BEOSIN
Blockchain Security

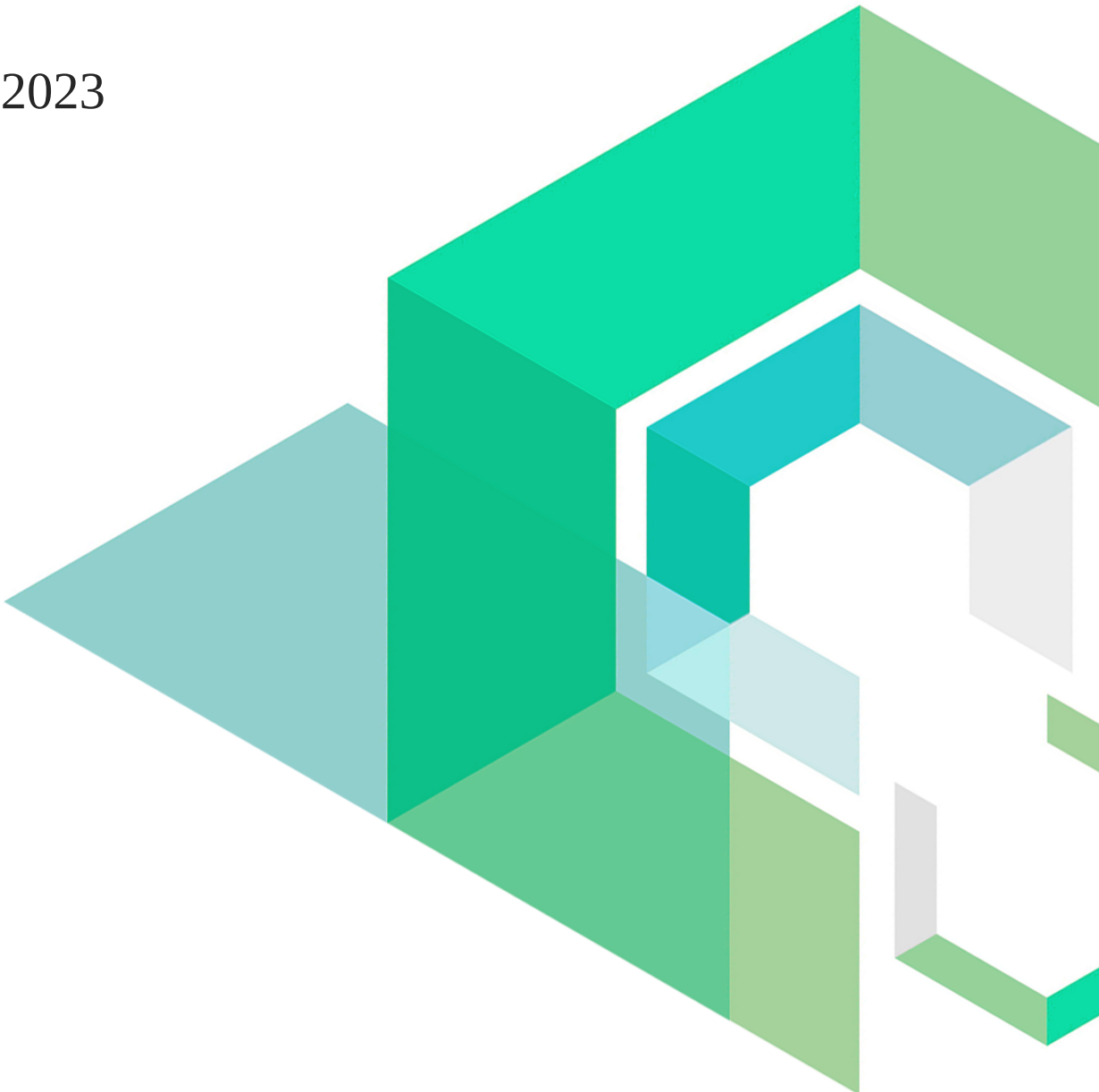
Minefi

Smart Contract Security Audit

V1.0

No. 202306120951

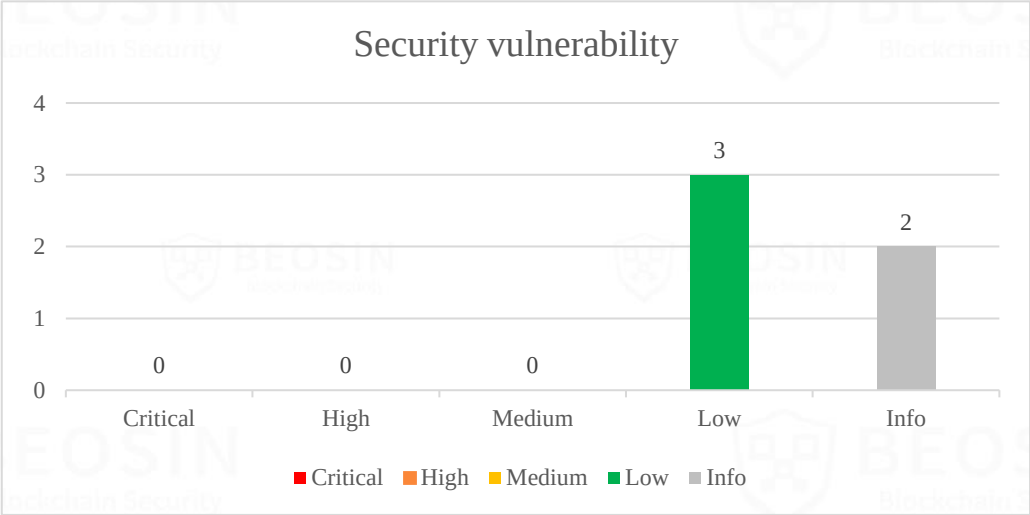
Jun 12th, 2023



Contents

Summary of Audit Results

After auditing, 3 Low risks and 2 Info items were identified in the Minefi project. Specific audit details will be presented in the **Findings** section. Users should pay attention to the following aspects when interacting with this project:



Project Description:

1. Business overview

MineFi is a decentralized storage revenue platform based on the FVM smart contract and dedicated to in-depth solutions to the security of pledged coins. Among them, the PledgeNFT1155V2 contract is related to the issuance of tokens, the redemption of collateral and the distribution of income. The Miner contract implements the logic of miner registration, confirms the logic of miner registration application, and processes various miner information. The Beneficiary contract realizes the recording of received FIL, management of the white list of NFT addresses, confirmation of changes in mining income distribution rights, and withdrawal Go to the specified wallet address, transfer, query and withdraw the balance of mining income, etc.

1 Overview

1.1 Project Overview

Project Name	Minefi
Platform	Filecoin
Audit Scope	Beneficiary.sol Miner.sol PledgeNFT1155V2.sol
File Hash	Beneficiary.sol bd98bc9013b933fad068a2f1d5375fa7901cd5c6289bc2b301f6a80956b82248 375c602688fb65ee7a5c669e69686474e6c4c9283e9adafae1dde08f3f7bd3ce Miner.sol eefad335b7cae24426efbc985d26ce67331dd238069100850b065c3bdc72e542 8f2a8fad687a454515e16ae77a5ff23aee8da3456d39d2a79be9a5dec4ef90ce PledgeNFT1155V2.sol 64d8e472425feeb71a22d05b926b19bae2828be6ef7cb7a8d81ac51a40622fe9 42a164b042a3656acd84b13ba6ff6b3fd2f6a7b9f298fde6ca2048eb84bed06c

1.2 Audit Overview

Audit work duration: May 23, 2023 –Jun 12, 2023

Audit methods: Formal Verification, Static Analysis, Typical Case Testing and Manual Review.

Audit team: Beosin Security Team.

2 Findings

Index	Risk description	Severity level	Status
Minefi – 1	There is a DoS attack vulnerability in *register*	Low	Fixed
Minefi – 2	Potential reentrancy risk in *mint*	Low	Fixed
Minefi – 3	The address authority in _mapNFTs is too large	Low	Acknowledged
Minefi – 4	Code redundancy	Info	Fixed
Minefi – 5	Missing event	Info	Fixed

Status Notes:

1. Minefi-3 is not fixed. The addresses in the _mapNFTs array can directly call the *transfer* function of the beneficiary contract to transfer the contract balance. The project party responded that the original design will be retained, and the nft address will be added after strict review by the owner, because when users receive benefits, they will need to make cross-contract transfers through the nft contract.

Finding Details:

[Minefi - 1] There is a DoS attack vulnerability in *register*

Severity Level	Low
Type	Business Security
Lines	Miner_audit.sol#L69-L84
Description	Attackers can preemptively submit the operation of registering miners, resulting in the occupation of the minerId and the administrator cannot delete it, which brings unnecessary trouble to the system.

```
function register(uint64 minerId, address minerAdmin, uint256 ransomRate) public
    require(
        _minerBase[minerId].state == MState.None,
        'Miner: The miner has been submitted for registration'
    );
    require(ransomRate > 0 && ransomRate <= 100, "Miner: Invalid data, the value can only range from
    _minerBase[minerId] = MinerBase(
        minerAdmin,
        address(0),
        0,
        MState.Audit,//更新状态
        block.timestamp,
        // DateTime.UTC8Timestamp(),
        ransomRate
    );
```

Figure 1 Source code of *register* function

Recommendations	It is recommended to add a method that can remove invalid minerIds.
Status	Fixed.

```
function register(uint64 minerId, address minerAdmin, uint256 ransomRate) public onlyOwner {
    require(
        _minerBase[minerId].state == MState.None,
        'Miner: The miner has been submitted for registration'
    );
    require(ransomRate > 0 && ransomRate <= 100, "Miner: Invalid data, the value can only range from 0 to 100");
    _minerBase[minerId] = MinerBase(
        minerAdmin,
        address(0),
        0,
        MState.Audit,
        block.timestamp,
        // DateTime.UTC8Timestamp(),
        ransomRate
    );
}
```

Figure 2 Source code of *register* function(fixed)

[Minefi - 2] Potential reentrancy risk in *mint*

Severity Level	**Low**
Type	Business Security
Lines	PledgeNFT1155V2_audit.sol #L404
Description	The *_mint* will call the *onERC1155Received* callback function that accepts the address. If the to address is a malicious contract, the *mint* function can be called again, although tokens still need to be paid, so it is a potential reentrancy risk.

```
        if (minNum == 0) {
            minNum = tokenNum;
        }
        _tokenIncomeStart[tokenNum] = startTime;
        _tokenIncomeEnd[tokenNum] = endTime;
        DS_MINT_COUNT.setStorageUint256(DS_MINT_COUNT.getStorageUint256() + 1);
        _tokenUser[tokenNum] = msg.sender;
    }

    _mint(msg.sender, META_TOKEN_ID.getStorageUint256(), count, "");

    payable(DS_PAYMENT_ADDRESS.getStorageAddress()).transfer(msg.value);

    emit MintEvent(msg.sender, META_TOKEN_ID.getStorageUint256(), minNum, _tokenNums.current(), startTime, endTime);
}
```

Figure 3 Source code of *mint* function

Recommendations	It is recommended to add a reentrant lock for the *mint* function.
Status	Fixed.

```
function mint(uint64 count) public payable nonReentrant {
    // require(getRemainCount() >= count, "oversold");
    require(CPS_PRICE.getStorageUint256() * count == msg.value, "amount error");
    // require(getMiner().getExceptionTerminateTime(CPS_MINER_ID.getStorageUint64()) == 0
    require(DS_PAYMENT_ADDRESS.getStorageAddress() != address(0), "recive wallet address

    uint256 curTime = block.timestamp;
    uint256 start = CPS_PACKING_DURATION.getStorageUint256() * DateTime.DAY_IN_SECONDS +
    uint256 end = start + DateTime.DAY_IN_SECONDS * (CPS_INCOME_DURATION.getStorageUint25
    uint256 startTime = DateTime.toTimestamp(
            start.getYear(),
            start.getMonth(),
            start.getDay(),
            0,0,0
```

Figure 4 Source code of *mint* function(fixed)

[Minefi - 3] The address authority in _mapNFTs is too large

Severity Level	**Low**
Type	Business Security
Lines	Beneficiary_audit.sol #L120-L25
Description	The addresses in the _mapNFTs array can directly call the *transfer* function of the beneficiary contract to transfer the contract balance.

```
/**
 * @param to: Transfer receiving address, the address cannot be zero
 * @param amount: Transfer amount
 * @dev Transfer a specific amount to a designated account
 */
function transfer(address to, uint256 amount) external nonReentrant {
    require(_mapNFTs[msg.sender], "access denied");
    require(to != address(0), "Transfer: The receiving address cannot be a zero address.");
    require(amount > 0 && amount <= address(this).balance, "Transfer: not sufficient funds");
    payable(to).transfer(amount);
}
```

Figure 5 Source code of *transfer* function

Recommendations	It is recommended that the owner be a multi-signature wallet.
Status	Acknowledged.

[Minefi - 4] Code redundancy

Severity Level	**Info**
Type	Coding Conventions
Lines	PledgeNFT1155V2_audit.sol #L637-L639
Description	The *_tokenExpire* function is not used.

```
function _tokenExpire(uint256 tokenNum) internal view returns (bool) {
    return (block.timestamp > _tokenIncomeStart[tokenNum]) && (block.timestamp <= _tokenIncomeEnd[tokenNum]);
}
```

Figure 6 Source code of *_tokenExpire* function

Recommendations	It is recommended to delete it directly.
Status	Fixed.

[Minefi - 5] Missing event

Severity Level	Info
Type	Coding Conventions
Lines	PledgeNFT1155V2_audit.sol#L142,L146,L155
Description	Many key operations in the contract do not add corresponding events, such as *setAdmin*, *delAdmin* and so on.

```
function setAdmin(address admin) public onlyOwner {
    _admins[admin] = true;
}

function delAdmin(address admin) public onlyOwner {
    delete _admins[admin];
}
```

Figure 7 Source code of *setAdmin* and *delAdmin* function

Recommendations	It is recommended to add corresponding events to key functions.
Status	Fixed.

```
function setAdmin(address admin) public onlyOwner {
    require(admin != address(0), "PledgeNFT: admin address cannot be zero");
    _admins[admin] = true;
    emit SetAdminEvent(admin);
}

/**
 * @dev if remove admin address, the event will be emitting
 * @param admin admin wallet address
 */
event RemoveAdminEvent(address admin);

function delAdmin(address admin) public onlyOwner {
    delete _admins[admin];
    emit RemoveAdminEvent(admin);
}
```

Figure 8 Source code of *setAdmin* and *delAdmin* function(fixed)

3 Appendix

3.1 Vulnerability Assessment Metrics and Status in Smart Contracts

3.1.1 Metrics

In order to objectively assess the severity level of vulnerabilities in blockchain systems, this report provides detailed assessment metrics for security vulnerabilities in smart contracts with reference to CVSS 3.1 (Common Vulnerability Scoring System Ver 3.1).

According to the severity level of vulnerability, the vulnerabilities are classified into four levels: "critical", "high", "medium" and "low". It mainly relies on the degree of impact and likelihood of exploitation of the vulnerability, supplemented by other comprehensive factors to determine of the severity level.

Likelihood \ Impact	Severe	High	Medium	Low
Probable	Critical	High	Medium	Low
Possible	High	High	Medium	Low
Unlikely	Medium	Medium	Low	Info
Rare	Low	Low	Info	Info

3.1.2 Degree of impact

- **Severe**

Severe impact generally refers to the vulnerability can have a serious impact on the confidentiality, integrity, availability of smart contracts or their economic model, which can cause substantial economic losses to the contract business system, large-scale data disruption, loss of authority management, failure of key functions, loss of credibility, or indirectly affect the operation of other smart contracts associated with it and cause substantial losses, as well as other severe and mostly irreversible harm.

- **High**

High impact generally refers to the vulnerability can have a relatively serious impact on the confidentiality, integrity, availability of the smart contract or its economic model, which can cause a greater economic loss, local functional unavailability, loss of credibility and other impact to the contract business system.

- **Medium**

Medium impact generally refers to the vulnerability can have a relatively minor impact on the confidentiality, integrity, availability of the smart contract or its economic model, which can cause a small amount of economic loss to the contract business system, individual business unavailability and other impact.

- **Low**

Low impact generally refers to the vulnerability can have a minor impact on the smart contract, which can pose certain security threat to the contract business system and needs to be improved.

3.1.4 Likelihood of Exploitation

- **Probable**

Probable likelihood generally means that the cost required to exploit the vulnerability is low, with no special exploitation threshold, and the vulnerability can be triggered consistently.

- **Possible**

Possible likelihood generally means that exploiting such vulnerability requires a certain cost, or there are certain conditions for exploitation, and the vulnerability is not easily and consistently triggered.

- **Unlikely**

Unlikely likelihood generally means that the vulnerability requires a high cost, or the exploitation conditions are very demanding and the vulnerability is highly difficult to trigger.

- **Rare**

Rare likelihood generally means that the vulnerability requires an extremely high cost or the conditions for exploitation are extremely difficult to achieve.

3.1.5 Fix Results Status

Status	Description
Fixed	The project party fully fixes a vulnerability.
Partially Fixed	The project party did not fully fix the issue, but only mitigated the issue.
Acknowledged	The project party confirms and chooses to ignore the issue.

3.2 Audit Categories

No.	Categories	Subitems
1	Coding Conventions	Compiler Version Security
		Deprecated Items
		Redundant Code
		require/assert Usage
		Gas Consumption
2	General Vulnerability	Integer Overflow/Underflow
		Reentrancy
		Pseudo-random Number Generator (PRNG)
		Transaction-Ordering Dependence
		DoS (Denial of Service)
		Function Call Permissions
		call/delegatecall Security
		Returned Value Security
		tx.origin Usage
		Replay Attack
		Overriding Variables
		Third-party Protocol Interface Consistency
3	Business Security	Business Logics
		Business Implementations
		Manipulable Token Price
		Centralized Asset Control
		Asset Tradability
		Minefitrage Attack

Beosin classified the security issues of smart contracts into three categories: Coding Conventions, General Vulnerability, Business Security. Their specific definitions are as follows:

- **Coding Conventions**

Audit whether smart contracts follow recommended language security coding practices. For example, smart contracts developed in Solidity language should fix the compiler version and do not use deprecated keywords.

- **General Vulnerability**

General Vulnerability include some common vulnerabilities that may appear in smart contract projects. These vulnerabilities are mainly related to the characteristics of the smart contract itself, such as integer overflow/underflow and denial of service attacks.

- **Business Security**

Business security is mainly related to some issues related to the business realized by each project, and has a relatively strong pertinence. For example, whether the lock-up plan in the code match the white paper, or the flash loan attack caused by the incorrect setting of the price acquisition oracle.

*Note that the project may suffer stake losses due to the integrated third-party protocol. This is not something Beosin can control. Business security requires the participation of the project party. The project party and users need to stay vigilant at all times.

3.3 Disclaimer

The Audit Report issued by Beosin is related to the services agreed in the relevant service agreement. The Project Party or the Served Party (hereinafter referred to as the "Served Party") can only be used within the conditions and scope agreed in the service agreement. Other third parties shall not transmit, disclose, quote, rely on or tamper with the Audit Report issued for any purpose.

The Audit Report issued by Beosin is made solely for the code, and any description, expression or wording contained therein shall not be interpreted as affirmation or confirmation of the project, nor shall any warranty or guarantee be given as to the absolute flawlessness of the code analyzed, the code team, the business model or legal compliance.

The Audit Report issued by Beosin is only based on the code provided by the Served Party and the technology currently available to Beosin. However, due to the technical limitations of any organization, and in the event that the code provided by the Served Party is missing information, tampered with, deleted, hidden or subsequently altered, the audit report may still fail to fully enumerate all the risks.

The Audit Report issued by Beosin in no way provides investment advice on any project, nor should it be utilized as investment suggestions of any type. This report represents an extensive evaluation process designed to help our customers improve code quality while mitigating the high risks in blockchain.

3.4 About Beosin

Beosin is the first institution in the world specializing in the construction of blockchain security ecosystem. The core team members are all professors, postdocs, PhDs, and Internet elites from world-renowned academic institutions. Beosin has more than 20 years of research in formal verification technology, trusted computing, mobile security and kernel security, with overseas experience in studying and collaborating in project research at well-known universities. Through the security audit and defense deployment of more than 2,000 smart contracts, over 50 public blockchains and wallets, and nearly 100 exchanges worldwide, Beosin has accumulated rich experience in security attack and defense of the blockchain field, and has developed several security products specifically for blockchain.

BEOSIN
Blockchain Security

Official Website

https://www.beosin.com

Telegram

https://t.me/+dD8Bnqd133RmNWNl

Twitter

https://twitter.com/Beosin_com

Email

Contact@beosin.com