BEOSIN
Blockchain Security

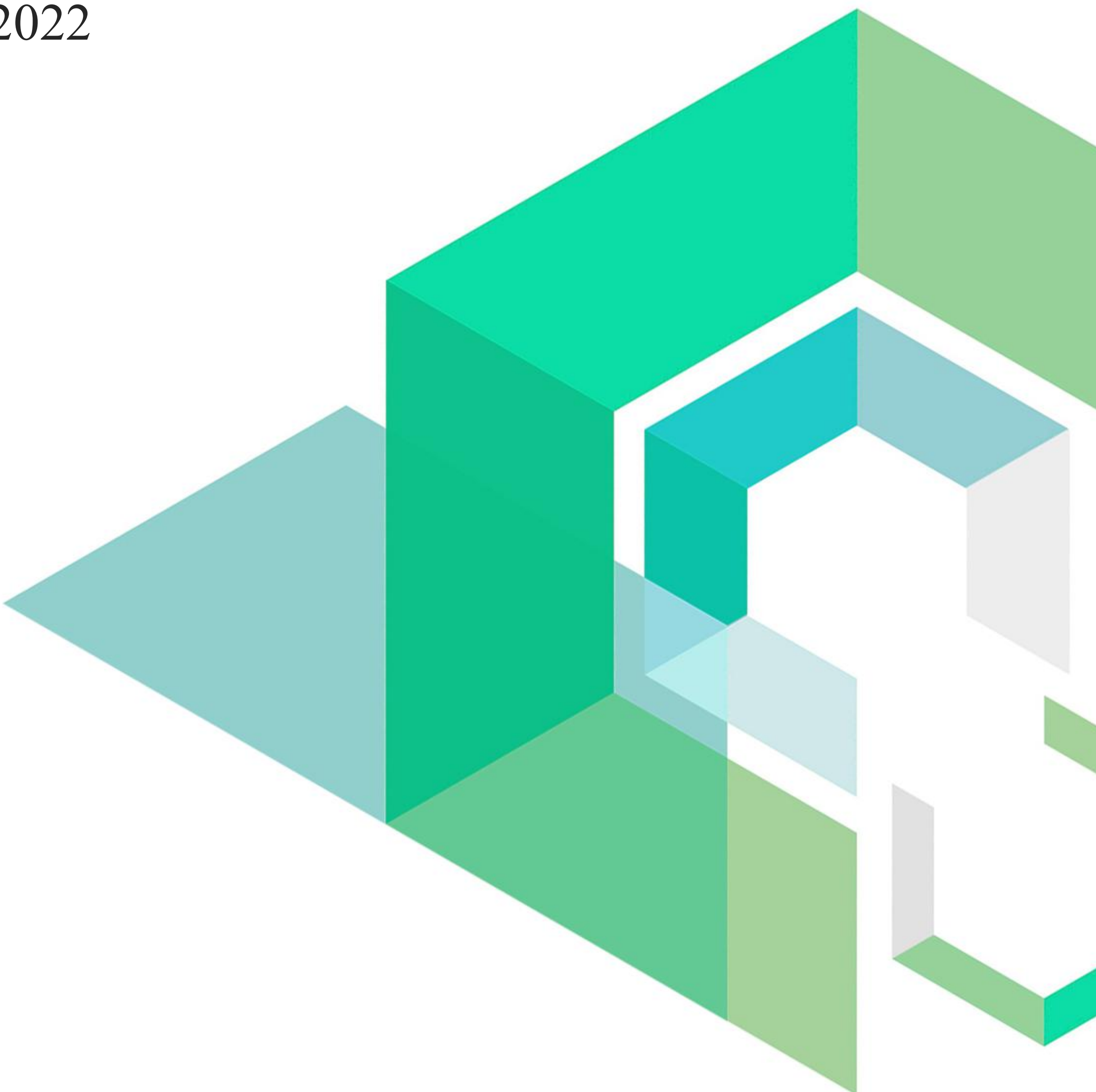
PEEL

Smart Contract Security Audit

V1.0

No. 202207151137

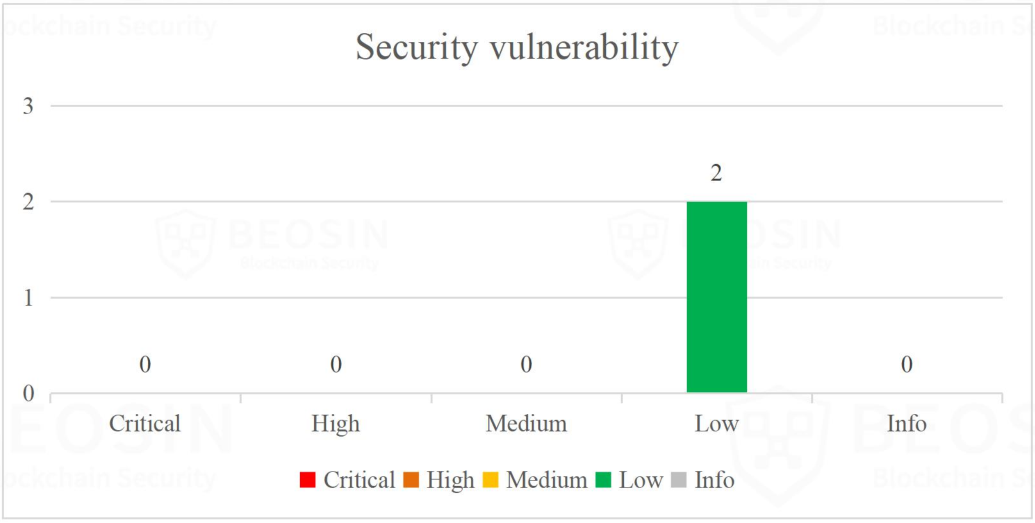
Jul 15th, 2022



Contents

Summary of audit results

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



*Notes:

- **Risk Description:**

  1. The swap limit set by the owner cannot be reached and the actual swap is always less than the cap.

  2. When the address burns its own tokens via the *burn* function, it does not use the internal function *_burnFrom*, the associated total will not be updated. The actual PEEL in circulation and the sum of all totals do not match.

- **Project Description:**

  1. Business overview

  PEEL is a ERC20 token. The owner of the contract can set the mint limit for a specified bridge address, and anyone can call *mint* to mint tokens to bridge address. The owner of the contract can also set the bridge token address and the swap cap of bridge tokens. The user can get PEEL by sending bridge tokens to this contract, or burn PEEL to get bridge tokens.

1 Overview

1.1 Project Overview

Project Name	PEEL
Platform	BNB Chain
Contract Address	0x734548a9e43d2D564600b1B2ed5bE9C2b911c6aB

1.2 Audit Overview

Audit work duration: July 12, 2022 – July 15, 2022

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

Audit team: Beosin Technology Co. Ltd.

2 Findings

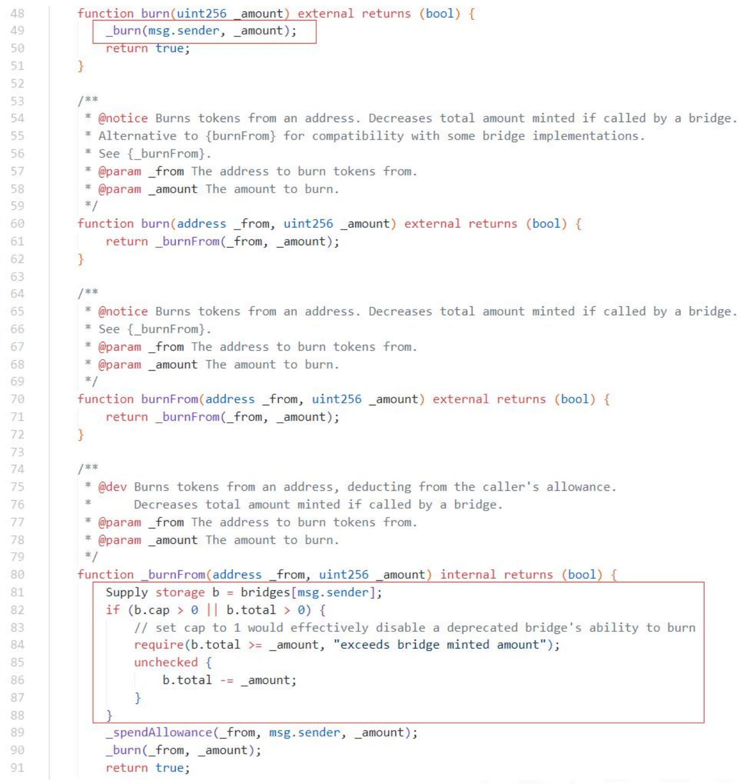
Index	Risk description	Severity level	Status
PEEL-1	Function implementation inconsistency	Low	Acknowledged
PEEL-2	Unable to reach swap limit	Low	Acknowledged

Status Notes:

- PEEL-1 is not fixed and may cause the actual PEEL in circulation and the sum of all totals do not match.
- PEEL-2 is not fixed and may be unable to reach the set swap limit.

[PEEL-1] Function implementation inconsistency

Severity Level	Low
Type	Business Security
Lines	MultiBridgeToken.sol#L48-91
Description	The implementation of *burn* in the MultiBridgeToken contract is different, and not using the internal function *_burnFrom* will lack the update of total.



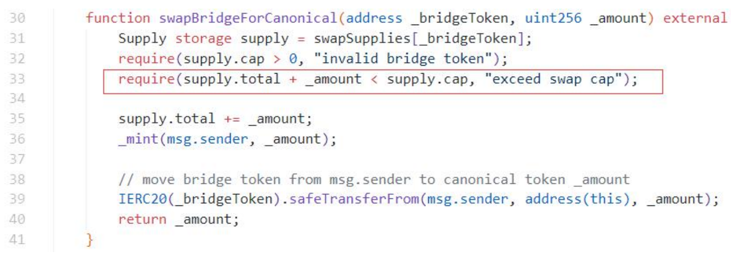
```
48  function burn(uint256 _amount) external returns (bool) {
49      _burn(msg.sender, _amount);
50      return true;
51  }
52
53  /**
54   * @notice Burns tokens from an address. Decreases total amount minted if called by a bridge.
55   * Alternative to {burnFrom} for compatibility with some bridge implementations.
56   * See {_burnFrom}.
57   * @param _from The address to burn tokens from.
58   * @param _amount The amount to burn.
59   */
60  function burn(address _from, uint256 _amount) external returns (bool) {
61      return _burnFrom(_from, _amount);
62  }
63
64  /**
65   * @notice Burns tokens from an address. Decreases total amount minted if called by a bridge.
66   * See {_burnFrom}.
67   * @param _from The address to burn tokens from.
68   * @param _amount The amount to burn.
69   */
70  function burnFrom(address _from, uint256 _amount) external returns (bool) {
71      return _burnFrom(_from, _amount);
72  }
73
74  /**
75   * @dev Burns tokens from an address, deducting from the caller's allowance.
76   *      Decreases total amount minted if called by a bridge.
77   * @param _from The address to burn tokens from.
78   * @param _amount The amount to burn.
79   */
80  function _burnFrom(address _from, uint256 _amount) internal returns (bool) {
81      Supply storage b = bridges[msg.sender];
82      if (b.cap > 0 || b.total > 0) {
83          // set cap to 1 would effectively disable a deprecated bridge's ability to burn
84          require(b.total >= _amount, "exceeds bridge minted amount");
85          unchecked {
86              b.total -= _amount;
87          }
88      }
89      _spendAllowance(_from, msg.sender, _amount);
90      _burn(_from, _amount);
91      return true;
```

Figure 1 Source code of related functions

Recommendations	It is recommended to change it to use *_burnFrom*.
Status	Acknowledged.

[PEEL-2] Unable to reach swap limit

Severity Level	Low
Type	Business Security
Lines	MintSwapCanonicalToken.sol#L33
Description	In *swapBridgeForCanonical* function, the judgment condition does not contain the case of equal, the number of swaps will not reach the set value.



```
function swapBridgeForCanonical(address _bridgeToken, uint256 _amount) external returns (uint256) {
    Supply storage supply = swapSupplies[_bridgeToken];
    require(supply.cap > 0, "invalid bridge token");
    require(supply.total + _amount < supply.cap, "exceed swap cap");

    supply.total += _amount;
    _mint(msg.sender, _amount);

    // move bridge token from msg.sender to canonical token _amount
    IERC20(_bridgeToken).safeTransferFrom(msg.sender, address(this), _amount);
    return _amount;
}
```

Figure 2 Source code of *swapBridgeForCanonical* function

Recommendations	It is recommended to change the judgment about the cap to <=.
Status	Acknowledged.

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.

Impact / Likelihood	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.Unfixed 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
		Arbitrage 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

Affiliated to BEOSIN Technology Pte. Ltd., 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