SALUS SECURITY

FEB 2023

# CODE SECURITY ASSESSMENT

META APES

# Overview

## Project Summary

- Name: META Apes
- Version: v1
- Platform: BNB Smart Chain
- Language: Solidity
- Contract address: 0x734548a9e43d2D564600b1B2ed5bE9C2b911c6aB
- Audit Range: See Appendix - 1

# Project Dashboard

## Application Summary

| | |
|---|---|
| **Name** | META Apes |
| **Version** | v1 |
| **Type** | Solidity |
| **Dates** | Feb 6 2023 |
| **Logs** | Feb 6 2023 |

## Vulnerability Summary

| | |
|---|---|
| **Total High-Severity issues** | 0 |
| **Total Medium-Severity issues** | 1 |
| **Total Low-Severity issues** | 1 |
| **Total informational issues** | 7 |
| **Total** | 9 |

## Contact

E-mail: support@salusec.io

# Risk Level Description

| | |
|---|---|
| **High Risk** | The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for clients' reputations or serious financial implications for clients and users. |
| **Medium Risk** | The issue puts a subset of users' sensitive information at risk, would be detrimental to the client's reputation if exploited, or is reasonably likely to lead to a moderate financial impact. |
| **Low Risk** | The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low impact in view of the client's business circumstances. |
| **Informational** | The issue does not pose an immediate risk, but is relevant to security best practices or defense in depth. |

# Content

# Introduction

## 1.1 About SALUS

At Salus Security, we are in the business of trust.

We are dedicated to tackling the toughest security challenges facing the industry today. By building foundational trust in technology and infrastructure through security, we help clients to lead their respective industries and unlock their full Web3 potential.

Our team of security experts employ industry-leading proof-of-concept (PoC) methodology for demonstrating smart contract vulnerabilities, coupled with advanced red teaming capabilities and a stereoscopic vulnerability detection service, to deliver comprehensive security assessments that allow clients to stay ahead of the curve.

In addition to smart contract audits and red teaming, our Rapid Detection Service for smart contracts aims to make security accessible to all. This high calibre, yet cost-efficient, security tool has been designed to support a wide range of business needs including investment due diligence, security and code quality assessments, and code optimisation.

We are reachable on Telegram (https://t.me/salusec), Twitter (https://twitter.com/salus_sec), or Email (support@salusec.io).

## 1.2 Audit Breakdown

The objective was to evaluate the repository for security-related issues, code quality, and adherence to specifications and best practices. Possible issues we looked for included (but are not limited to):

- Risky external calls
- Integer overflow/underflow
- Transaction-ordering dependence
- Timestamp dependence
- Access control
- Call stack limits and mishandled exceptions
- Number rounding errors
- Centralization of power
- Logical oversights and denial of service
- Business logic specification
- Code clones, functionality duplication

## 1.3 Disclaimer

Note that this security audit is not designed to replace functional tests required before any software release and does not give any warranties on finding all possible security issues with the given smart contract(s) or blockchain software, i.e., the evaluation result does not guarantee the nonexistence of any further findings of security issues.

# Findings

## 2.1 Summary of Findings

| ID | Title | Severity | Category | Status |
|---|---|---|---|---|
| 1 | Centralization risk | Medium | Centralization | Unresolved |
| 2 | Incompatible with fee-on-transfer tokens and rebase token | Low | Business Logic | Unresolved |
| 3 | Mismatch between comment and implementation | Informational | Business Logic | Unresolved |
| 4 | Can use OpenZeppelin's Ownable2Step to replace Ownable | Informational | Authentication | Unresolved |
| 5 | Using outdated OpenZeppelin library version | Informational | Configuration | Unresolved |
| 6 | Floating compiler version | Informational | Configuration | Unresolved |
| 7 | Missing events for critical functions | Informational | Auditing and Logging | Unresolved |
| 8 | Indexed modifier can be added to event parameters | Informational | Auditing and Logging | Unresolved |
| 9 | Gas optimization suggestions | Informational | Gas optimization | Unresolved |

## 2.2 Notable Findings

Significant flaws that impact system confidentiality, integrity, or availability are listed below.

### 1. Centralization risk

| Severity: Medium | Category: Centralization |
| --- | --- |

Target:
- contracts\pegged-bridge\tokens\MultiBridgeToken.sol
- contracts\pegged-bridge\tokens\MintSwapCanonicalToken.sol

### Description

**contracts\pegged-bridge\tokens\MultiBridgeToken.sol:L106-L110**

```
function updateBridgeSupplyCap(address _bridge, uint256 _cap) external onlyOwner {
    // cap == 0 means revoking bridge role
    bridges[_bridge].cap = _cap;
    emit BridgeSupplyCapUpdated(_bridge, _cap);
}
```

**contracts\pegged-bridge\tokens\MintSwapCanonicalToken.sol:L65-L68**

```
function setBridgeTokenSwapCap(address _bridgeToken, uint256 _swapCap) external
onlyOwner {
    swapSupplies[_bridgeToken].cap = _swapCap;
    emit TokenSwapCapUpdated(_bridgeToken, _swapCap);
}
```

The **updateBridgeSupplyCap** function in the contract is restricted to be called only by the Owner role. If the owner's private key is compromised and obtained by a malicious person, then he or she can set a large cap value for his own address by calling this function, and then call the **mint** function to complete a large amount of token issuance.

Moreover, the owner role can also call the **setBridgeTokenSwapCap** function to change the value of `swapSupplies[_bridgeToken].cap`, if the owner's private key is compromised and obtained by a malicious person who then call this function to change the cap to 0, other users will not be able to swap tokens by using the **swapBridgeForCanonical** and **swapCanonicalForBridge** functions.

### Recommendation

It is recommended to use a Multisig wallet for the owner address.

## 2. Incompatible with fee-on-transfer tokens and rebase token

| Severity: Low | Category: Business Logic |
| --- | --- |

Target:
- contracts\pegged-bridge\tokens\MintSwapCanonicalToken.sol

## Description

**contracts\pegged-bridge\tokens\MintSwapCanonicalToken.sol:L25-L57**

```
/**
 * @notice msg.sender has bridge token and wants to get canonical token.
 * @param _bridgeToken The intermediary token address for a particular bridge.
 * @param _amount The amount.
 */
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;
}

/**
 * @notice msg.sender has canonical token and wants to get bridge token (eg. for cross
chain burn).
 * @param _bridgeToken The intermediary token address for a particular bridge.
 * @param _amount The amount.
 */
function swapCanonicalForBridge(address _bridgeToken, uint256 _amount) external returns
(uint256) {
    Supply storage supply = swapSupplies[_bridgeToken];
    require(supply.cap > 0, "invalid bridge token");

    supply.total -= _amount;
    _burn(msg.sender, _amount);

    IERC20(_bridgeToken).safeTransfer(msg.sender, _amount);
    return _amount;
}
```

The correct functioning of the **swapCanonicalForBridge** and **swapBridgeForCanonical** functions in the contract depends on the corresponding **_bridgeToken** being a standard ERC20 token. If the token is a fee-on-transfer token or a rebase token, the user may not be able to swap it back using the **swapCanonicalForBridge** function after swapping the **MultiBridgeToken** using the **swapCanonicalForBridge** function because the number of **_bridgeTokens** in the contract may be less than the expected value.

## Recommendation

We recommend making sure **_bridgeToken** is a standard ERC20 token.

# 2.3 Informational Findings

| 3. Mismatch between comment and implementation | |
|---|---|
| Severity: Informational | Category: Business Logic |
| Target:<br>- contracts\pegged-bridge\tokens\MultiBridgeToken.sol | |

## Description

**contracts\pegged-bridge\tokens\MultiBridgeToken.sol:L64-L92**

```
/**
 * @notice Burns tokens from an address. Decreases total amount minted if called by a bridge.
 * See {_burnFrom}.
 * @param _from The address to burn tokens from.
 * @param _amount The amount to burn.
 */
function burnFrom(address _from, uint256 _amount) external returns (bool) {
    return _burnFrom(_from, _amount);
}

/**
 * @dev Burns tokens from an address, deducting from the caller's allowance.
 *      Decreases total amount minted if called by a bridge.
 * @param _from The address to burn tokens from.
 * @param _amount The amount to burn.
 */
function _burnFrom(address _from, uint256 _amount) internal returns (bool) {
    Supply storage b = bridges[msg.sender];
    if (b.cap > 0 || b.total > 0) {
        // set cap to 1 would effectively disable a deprecated bridge's ability to burn
        require(b.total >= _amount, "exceeds bridge minted amount");
        unchecked {
            b.total -= _amount;
        }
    }
    _spendAllowance(_from, msg.sender, _amount);
    _burn(_from, _amount);
    return true;
}
```

The code comment states that the **burnFrom** function `'Decreases the total amount minted if called by a bridge'`, but according to its code implementation, the decrease can be made when **burnFrom()** is called by any user approved address.

## Recommendation

It is recommended to fix the comment so that it matches with the implementation.

## 4. Can use OpenZeppelin’s Ownable2Step to replace Ownable

| Severity: Informational | Category: Authentication |
| --- | --- |
| Target:<br>- contracts\safeguard\Ownable.sol | |

### Description

The project uses a custom Ownable.sol contract. It is recommended to use OpenZeppelin’s Ownable contract as it is well tested and optimized.

There is also an Ownable2Step.sol contract in the OpenZeppelin library that is designed for two-step ownership transfer which is more secure than the single-step ownership transfer used by Ownable.sol.

### Recommendation

Consider use OpenZeppelin's Ownable2Step.sol instead of Ownable.sol.

## 5. Using outdated OpenZeppelin library version

| Severity: Informational | Category: Configuration |
| --- | --- |
| Target:<br>- contracts\pegged-bridge\tokens\MultiBridgeToken.sol<br>- contracts\pegged-bridge\tokens\MintSwapCanonicalToken.sol | |

### Description

The contract uses version 4.5.0 for its OpenZeppelin's dependencies, but this version has a High severity vulnerability related to ECDSA - Reference. Even though the code is not exploitable in its current state, it is best to upgrade the OpenZeppelin library dependency to the latest safe version to get all security patches, features and gas optimisations.

### Recommendation

Consider using the latest safe version of the OpenZeppelin library.

## 6. Floating compiler version

| Severity: Informational | Category: Configuration |
| --- | --- |
| Target:<br>- all | |

## Description

```
pragma solidity ^0.8.9;
```

The META Apes contracts use a floating compiler version ^0.8.9.

Using a floating pragma ^0.8.9 statement is discouraged, as code may compile to different bytecodes with different compiler versions. Use a locked pragma statement to get a deterministic bytecode. Also use the latest Solidity version to get all the compiler features, bug fixes and optimizations.

## Recommendation

It is recommended to use a locked Solidity version throughout the project. It is also recommended to use the most stable and up-to-date version.

## 7. Missing events for critical functions

| Severity: Informational | Category: Auditing and Logging |
| --- | --- |

Target:
- contracts\pegged-bridge\tokens\MultiBridgeToken.sol
- contracts\pegged-bridge\tokens\MintSwapCanonicalToken.sol

## Description

**contracts\pegged-bridge\tokens\MintSwapCanonicalToken.sol:L26-L57**

```
/**
 * @notice msg.sender has bridge token and wants to get canonical token.
 * @param _bridgeToken The intermediary token address for a particular bridge.
 * @param _amount The amount.
 */
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;
}

/**
 * @notice msg.sender has canonical token and wants to get bridge token (eg. for cross chain burn).
 * @param _bridgeToken The intermediary token address for a particular bridge.
 * @param _amount The amount.
 */
function swapCanonicalForBridge(address _bridgeToken, uint256 _amount) external returns (uint256) {
    Supply storage supply = swapSupplies[_bridgeToken];
    require(supply.cap > 0, "invalid bridge token");

    supply.total -= _amount;
    _burn(msg.sender, _amount);

    IERC20(_bridgeToken).safeTransfer(msg.sender, _amount);
    return _amount;
}
```

**swapCanonicalForBridge** function and **swapBridgeForCanonical** function are important functions that miss event emission. Relevant events should be added to ensure that changes in such functions can be tracked.

## Recommendation

Consider adding event emission to **swapCanonicalForBridge** function and **swapBridgeForCanonical** function.

## 8. Indexed modifier can be added to event parameters

| Severity: Informational | Category: Auditing and Logging |
| --- | --- |

Target:
- contracts\pegged-bridge\tokens\MultiBridgeToken.sol
- contracts\pegged-bridge\tokens\MintSwapCanonicalToken.sol

## Description

**contracts\pegged-bridge\tokens\MultiBridgeToken.sol:L20**

```
event BridgeSupplyCapUpdated(address bridge, uint256 supplyCap);
```

**contracts\pegged-bridge\tokens\MintSwapCanonicalToken.sol:L17**

```
event TokenSwapCapUpdated(address token, uint256 cap);
```

Indexed fields in events make it easy for the off-chain indexer to track the fields. If there are fewer than three fields, all of the fields should be indexed.  But also note that each indexed field costs extra gas.

## Recommendation

We recommend that you add indexed modifiers to fields in event **TokenSwapCapUpdated** and event **BridgeSupplyCapUpdated.**

## 9. Gas optimization suggestions

| Severity: Informational | Category: gas optimization |
| --- | --- |

Target:
- contracts\pegged-bridge\tokens\MultiBridgeToken.sol

## Description

**contracts\pegged-bridge\tokens\MultiBridgeToken.sol:L37**

```
require(b.cap > 0, "invalid caller");
```

The contract uses the 'require' statement for a conditional check, but starting from Solidity v0.8.4, there is a convenient and gas-efficient way to explain to users why an operation failed through the use of **custom errors**. Using custom errors can reduce both gas usage and bytecode size.

**contracts\pegged-bridge\tokens\MultiBridgeToken.sol:L97**

```
function decimals() public view virtual override returns (uint8)
```

The **decimals()** function is not called inside the contract, thus, the visibility can be changed to external to save gas.

## Recommendation

Consider using custom errors to replace the revert messages.

Consider changing the visibility of the decimals() function to external.

# Appendix

## Appendix 1 - Files in Scope

This audit covered the following files:

| File | SHA-1 hash |
|---|---|
| MintSwapCanonicalToken.sol | 465f20faa94e50ef745c39b693a85d7ebf350f00 |
| MultiBridgeToken.sol | d8c8ecb01dd46651f0500d872fbc188b8be6e522 |
| Ownable.sol | 09d272559cb58aa22cd74780b9465ffcd6d4693d |