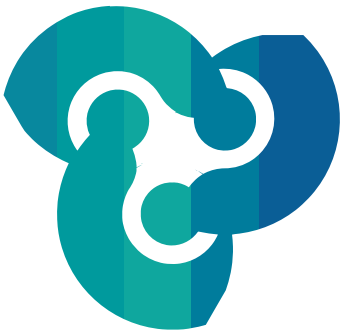
Serial number: 202203310496



Artemis

Security Specialist

Security Audit Report
Smart Contract

Artemis cloud company

2022 / 3 / 31

PUBLISHED INFORMATION

REPORT NUMBER	202203310496
DATE	2022-03-31

PROJECT INFORMATION

TITLE	MoonbaSwap
SYMBOL	MoonbaSwap
CONTRACT ADDRESS	https://www.moonba.co

VULNERABILITY ANALYSIS

HIGH	0	No high risk vulnerability spotted.
MEDIUM	0	No medium risk vulnerability spotted.
LOW	0	No low risk vulnerability spotted.
RECOMMENDATION	5	5 enhanced suggestions

Contents

1、executive summary

March 31, 2022, the Artemis Cloud Company security team received the Moonba0.8.4, 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 Artemis Cloud Company 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.

Artemis Cloud Company Smart Contract MinT project test method:

Black box testing	Conduct security tests from an attacker's perspective externally.
Grey box testing	On the basis of the project white list, the project is carefully tested and the possible loopholes are excavated.
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.

Artemis Cloud Company Smart Contract DeFi project risk level:

Critical vulnerabilities	Critical vulnerabilities will have a significant impact on the security of the DeFi project, and it is strongly recommended to fix the critical vulnerabilities.
High-risk vulnerabilities	High-risk vulnerabilities will affect the normal operation of DeFi project. It is strongly recommended to fix high-risk vulnerabilities.
Medium-risk vulnerabilities	Medium vulnerability will affect the operation of DeFi project. It is recommended to fix medium-risk vulnerabilities.
Low-risk vulnerabilities	Low-risk vulnerabilities may affect the operation of DeFi project in certain scenarios. It is suggested that the project party should evaluate and consider whether these vulnerabilities need to be fixed.

Weaknesses	There are safety risks theoretically, but it is extremely difficult to reproduce in engineering.
Enhancement Suggestions	There are better practices for coding or architecture.

2、Audit Methodology

Our security audit process for smart contract includes two steps:

1、 Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using public and in-house automated analysis tools.

2、 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:

- Reentrancy attack and other Race Conditions
- Replay attack
- Reordering attack
- Short address attack
- Denial of service attack
- Transaction Ordering Dependence attack
- Conditional Completion attack
- Authority Control attack
- Integer Overflow and Underflow attack
- Timestamp Dependence attack
- Gas Usage, Gas Limit and Loops
- Redundant fallback function
- Unsafe type Inference
- Explicit visibility of functions state variables
- Logic Flaws
- Uninitialized Storage Pointers
- Floating Points and Numerical Precision
- tx.origin Authentication
- "False top-up" Vulnerability
- Scoping and Declarations

3、Project Background

3.1 Project Introduction

Moonba is a DEX built on the OneLedger platform that allows trading of multiple tokens with no intervention; effectively eliminating the middleman. It is created specifically for the OneLedger ecosystem and the future of interoperability. It strives for liquidity, security, convenience, and a user -friendly interface that allows for an immediate asset exchange.

Contract Name:

MoonbaSwap

Audit version file information:

merge.sol

File(s) Hash:

merge.sol(SHA256): 8da2ea088c9b62791a1649563e4b3f50fc8c718baf2b69162a88750deaa05b3d

Contract Address:

https://www.moonba.co

3.2 Project Structure

-- interfaces/

-- BridgeRouter.sol

├── IBridgeCosignerManager.sol

├── IBridgeRouter.sol

├── IBridgeToken.sol

├── IBridgeTokenManager.sol

├── IOwnable.sol

└── IWETH.sol

-- library/

├── RLPReader.sol

└── RToken.sol

-- managers/

├── BridgeCosignerManager.sol

└── BridgeTokenManager.sol

-- tokens/

└── BridgeToken.sol

-- versions/

├── Version0.sol

└── Version1.sol

4、Code Overview

4.1 Contracts Description

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

merge.sol			
Function Name	Visibility	Mutability	Modifiers
toString	Internal	-	-
toHexString	Internal	-	-
toHexString	Internal	-	-
_throwError	Private	-	-
tryRecover	Internal	-	-
recover	Internal	-	-
tryRecover	Internal	-	-
recover	Internal	-	-
tryRecover	Internal	-	-
recover	Internal	-	-
toEthSignedMessageHash	Internal	-	-
toEthSignedMessageHash	Internal	-	-
toTypedDataHash	Internal	-	-
isContract	Internal	-	-
sendValue	Internal	Can modify state	-
functionCall	Internal	Can modify state	-
functionCall	Internal	Can modify state	-
functionCallWithValue	Internal	Can modify state	-
functionCallWithValue	Internal	Can modify state	-
functionStaticCall	Internal	-	-
functionStaticCall	Internal	-	-
functionDelegateCall	Internal	Can modify state	-
functionDelegateCall	Internal	Can modify state	-

Function Name	Visibility	Mutability	Modifiers
verifyCallResult	Internal	-	-
safeTransfer	Internal	Can modify state	-
safeTransferFrom	Internal	Can modify state	-
safeApprove	Internal	Can modify state	-
safeIncreaseAllowance	Internal	Can modify state	-
safeDecreaseAllowance	Internal	Can modify state	-
_callOptionalReturn	Private	Can modify state	-
unsafeTransfer	Internal	Can modify state	-
enter	Internal	Can modify state	-
exit	Internal	Can modify state	-
next	Internal	-	-
hasNext	Internal	-	-
toRlpItem	Internal	-	-
iterator	Internal	-	-
rlpLen	Internal	-	-
payloadLocation	Internal	-	-
payloadLen	Internal	-	-
toList	Internal	-	-
isList	Internal	-	-
rlpBytesKeccak256	Internal	-	-
payloadKeccak256	Internal	-	-
toRlpBytes	Internal	-	-
toBoolean	Internal	-	-
toAddress	Internal	-	-
toUint	Internal	-	-
toUintStrict	Internal	-	-
toBytes	Internal	-	-
numItems	Private	-	-
_itemLength	Private	-	-
_payloadOffset	Private	-	-
copy	Private	-	-

Function Name	Visibility	Mutability	Modifiers
isContract	Internal	-	-
sendValue	Internal	Can modify state	-
functionCall	Internal	Can modify state	-
functionCall	Internal	Can modify state	-
functionCallWithValue	Internal	Can modify state	-
functionCallWithValue	Internal	Can modify state	-
functionStaticCall	Internal	-	-
functionStaticCall	Internal	-	-
verifyCallResult	Internal	-	-
totalSupply	External	-	-
balanceOf	External	-	-
transfer	External	Can modify state	-
allowance	External	-	-
approve	External	Can modify state	-
transferFrom	External	Can modify state	-
name	External	-	-
symbol	External	-	-
decimals	External	-	-
permit	External	Can modify state	-
nonces	External	-	-
DOMAIN_SEPARATOR	External	-	-
totalSupply	External	-	-
balanceOf	External	-	-
transfer	External	Can modify state	-
allowance	External	-	-
approve	External	Can modify state	-
transferFrom	External	Can modify state	-
transferOwnership	External	Can modify state	-
deposit	External	payable	-
withdraw	External	Can modify state	-
addCosigner	External	Can modify state	-

Function Name	Visibility	Mutability	Modifiers
addCosignerBatch	External	Can modify state	-
removeCosigner	External	Can modify state	-
removeCosignerBatch	External	Can modify state	-
getCosigners	External	-	-
getCosignCount	External	-	-
verify	External	-	-
enter	External	Can modify state	-
enterETH	External	payable	-
exit	External	Can modify state	-
burn	External	Can modify state	-
mint	External	Can modify state	-
updateTokenInfo	External	Can modify state	-
issue	External	Can modify state	-
revoke	External	Can modify state	-
getLocal	External	-	-
isZero	External	-	-
supportsInterface	External	-	-
_msgSender	Internal	-	-
_msgData	Internal	-	-
<Constructor>	Public	Can modify state	-
owner	Public	-	-
renounceOwnership	Public	Can modify state	onlyOwner
transferOwnership	Public	Can modify state	onlyOwner
_transferOwnership	Internal	Can modify state	-
supportsInterface	Public	-	-
_disableInitializers	Internal	Can modify state	-
_setInitializedVersion	Private	Can modify state	-
__Context_init	Internal	Can modify state	onlyInitializing
__Context_init_unchained	Internal	Can modify state	onlyInitializing
_msgSender	Internal	-	-
_msgData	Internal	-	-

Function Name	Visibility	Mutability	Modifiers
__Pausable_init	Internal	Can modify state	onlyInitializing
__Pausable_init_unchained	Internal	Can modify state	onlyInitializing
paused	Public	-	-
_pause	Internal	Can modify state	whenNotPaused
_unpause	Internal	Can modify state	whenPaused
__ReentrancyGuard_init	Internal	Can modify state	onlyInitializing
__ReentrancyGuard_init_unchained	Internal	Can modify state	onlyInitializing
__Ownable_init	Internal	Can modify state	onlyInitializing
__Ownable_init_unchained	Internal	Can modify state	onlyInitializing
owner	Public	-	-
renounceOwnership	Public	Can modify state	onlyOwner
transferOwnership	Public	Can modify state	onlyOwner
_transferOwnership	Internal	Can modify state	-
<Fallback>	External	payable	-
emitEnter	Internal	Can modify state	-
emitExit	Internal	Can modify state	-
setTokenManager	External	Can modify state	onlyOwner
setCosignerManager	External	Can modify state	onlyOwner
updateTokenInfo	External	Can modify state	onlyOwner
transferTokenOwnership	External	Can modify state	onlyOwner
initialize	Public	Can modify state	initializer
enter	External	Can modify state	nonReentrant
enterETH	External	payable	nonReentrant
exit	External	Can modify state	nonReentrant
__ERC20_init	Internal	Can modify state	onlyInitializing
__ERC20_init_unchained	Internal	Can modify state	onlyInitializing
name	Public	-	-
symbol	Public	-	-
decimals	Public	-	-
totalSupply	Public	-	-
balanceOf	Public	-	-

Function Name	Visibility	Mutability	Modifiers
transfer	Public	Can modify state	-
allowance	Public	-	-
approve	Public	Can modify state	-
transferFrom	Public	Can modify state	-
increaseAllowance	Public	Can modify state	-
decreaseAllowance	Public	Can modify state	-
_transfer	Internal	Can modify state	-
_mint	Internal	Can modify state	-
_burn	Internal	Can modify state	-
_approve	Internal	Can modify state	-
_spendAllowance	Internal	Can modify state	-
_beforeTokenTransfer	Internal	Can modify state	-
_afterTokenTransfer	Internal	Can modify state	-
initialize	Public	Can modify state	initializer
updateTokenInfo	External	Can modify state	onlyOwner
mint	External	Can modify state	onlyOwner
burn	External	Can modify state	onlyOwner
name	Public	-	-
symbol	Public	-	-
decimals	Public	-	-
nonces	External	-	-
permit	External	Can modify state	-
DOMAIN_SEPARATOR	Public	-	-
toString	Internal	-	-
<Constructor>	Public	Can modify state	-
supportsInterface	Public	-	-
getLocal	Public	-	-
isZero	Public	-	-
revoke	External	Can modify state	onlyOwner
issue	External	Can modify state	onlyOwner
createKey	Private	-	-

Function Name	Visibility	Mutability	Modifiers
supportsInterface	Public	-	-
addCosigner	Public	Can modify state	onlyOwner
addCosignerBatch	Public	Can modify state	onlyOwner
removeCosigner	Public	Can modify state	onlyOwner
removeCosignerBatch	Public	Can modify state	onlyOwner
getCosigners	Public	-	-
getCosignCount	Public	-	-
recover	Internal	-	-
verify	External	-	-
_inCache	Internal	-	-

4.2 Code Audit

4.2.1 HIGH-risk vulnerabilities

N/A

4.2.2 Medium-risk vulnerabilities

N/A

4.2.3 LOW-risk vulnerabilities

N/A

4.2.4 Enhanced recommendations

4.2.4.1A floating pragma is set.

The current pragma Solidity directive is ""^0.8.4"". It is recommended to specify a fixed compiler version to ensure that the bytecode produced does not vary between builds. This is especially important if you rely on bytecode-level verification of the code.

Code location:

merge.sol file,Line 2

```
1  // SPDX-License-Identifier: MIT
2  pragma solidity ^0.8.4;
```

```
3

4   library Strings {
```

4.2.4.2Potential use of "blockhash" as source of randonmness.

The environment variable "blockhash" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Code location:

merge.sol file,Line 2717

```
2715    _salt = keccak256(
2716    abi.encodePacked(
2717     blockhash(block.number - 1),
2718     block.timestamp,
2719    block.difficulty,
2720    block.coinbase
```

4.2.4.3Potential use of "block.number" as source of randonmness.

The environment variable "block.number" looks like it might be used as a source of randomness. Note that the values of variables like coinbase, gaslimit, block number and timestamp are predictable and can be manipulated by a malicious miner. Also keep in mind that attackers know hashes of earlier blocks. Don't use any of those environment variables as sources of randomness and be aware that use of these variables introduces a certain level of trust into miners.

Code location:

merge.sol file,Line 2717

```
2715    _salt = keccak256(
2716    abi.encodePacked(
```

```
2717    blockhash(block.number - 1),
2718    block.timestamp,
2719    block.difficulty,
```

4.2.4.4Requirement violation.

A requirement was violated in a nested call and the call was reverted as a result. Make sure valid inputs are provided to the nested call (for instance, via passed arguments).

Code location:

merge.sol file,Line 2094

```
2092    {
2093    require(msg.value != 0, "BR: ZERO_AMOUNT");
2094    require(tokenManager.isZero(targetChainId), "BR: NOT_FOUND");
2095
2096    emitEnter(address(0), _msgSender(), msg.value, targetChainId);
```

4.2.4.5Requirement violation.

A requirement was violated in a nested call and the call was reverted as a result. Make sure valid inputs are provided to the nested call (for instance, via passed arguments).

Code location:

merge.sol file,Line 2050 merge.sol file,Line 1882

```
2048    onlyOwner
2049    {
2050    IOwnable(token).transferOwnership(newOwner);
2051    }
2052
2053    // Initialize function for proxy constructor. Must be used atomically
1880    */
```

```
1881    modifier onlyOwner() {
1882     require(owner() == _msgSender(), "Ownable: caller is not the owner");
1883     _;
1884    }
1885
```

5、Audit Result

5.1 Conclusion

Audit Result:

Enhanced recommendations are known to exist

Audit Number:

202203310496

Audit Date:

2022-03-31

Audit Team:

Artemis Security Team

Summary conclusion:

HIGH	0	No high risk vulnerability spotted.
MEDIUM	0	No medium risk vulnerability spotted.
LOW	0	No low risk vulnerability spotted.
RECOMMENDATION	5	5 enhanced suggestions

6、Statement

Artemis Cloud Company issues this report with reference to the facts that have occurred or existed before the issuance of this report, and only assumes corresponding responsibility base on these. For the facts that occurred or existed after the issuance, Artemis Cloud Company 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 Artemis Cloud Company by the information provider till the date of the insurance this report (referred to as "provided information"). Artemis Cloud Company 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 Artemis Cloud Company shall not be liable for any loss or adverse effect resulting therefrom. Artemis Cloud Company only conducts the agreed security audit on the security situation of the project and issues this report. Artemis Cloud Company is not responsible for the background and other conditions of the project.

Website

https://www.artemisclouds.com/

E-mail

sales@artemisclouds.com