verichains

SECURITY AUDIT OF

U2U BRIDGE SMART CONTRACT

Unicorn Ultra

Public Report

Sep 25, 2023

Verichains Lab

info@verichains.io

https://www.verichains.io

Driving Technology > Forward

ABBREVIATIONS

Name	Description
Ethereum	An open source platform based on blockchain technology to create and distribute smart contracts and decentralized applications.
Ether (ETH)	A cryptocurrency whose blockchain is generated by the Ethereum platform. Ether is used for payment of transactions and computing services in the Ethereum network.
Smart contract	A computer protocol intended to digitally facilitate, verify or enforce the negotiation or performance of a contract.
Solidity	A contract-oriented, high-level language for implementing smart contracts for the Ethereum platform.
Solc	A compiler for Solidity.
ERC20	ERC20 (BEP20 in Binance Smart Chain or *x*RP20 in other chains) tokens are blockchain-based assets that have value and can be sent and received. The primary difference with the primary coin is that instead of running on their own blockchain, ERC20 tokens are issued on a network that supports smart contracts such as Ethereum or Binance Smart Chain.

EXECUTIVE SUMMARY

This Security Audit Report was prepared by Verichains Lab on Sep 25, 2023. We would like to thank the Unicorn Ultra (U2U) for trusting Verichains Lab in auditing smart contracts. Delivering high-quality audits is always our top priority.

This audit focused on identifying security flaws in code and the design of the U2U Bridge Smart Contract. The scope of the audit is limited to the source code files provided to Verichains. Verichains Lab completed the assessment using manual, static, and dynamic analysis techniques.

During the audit process, the audit team had identified some vulnerable issues in the contract code.

TABLE OF CONTENTS

1. MANAGEMENT SUMMARY

1.1. About U2U Bridge Smart Contract

U2U Bridge is a cross-chain infrastructure that allows for the simple movement of crypto assets, tokens, and data to other compatible blockchain networks. The U2U team describes the bridging process as follows:

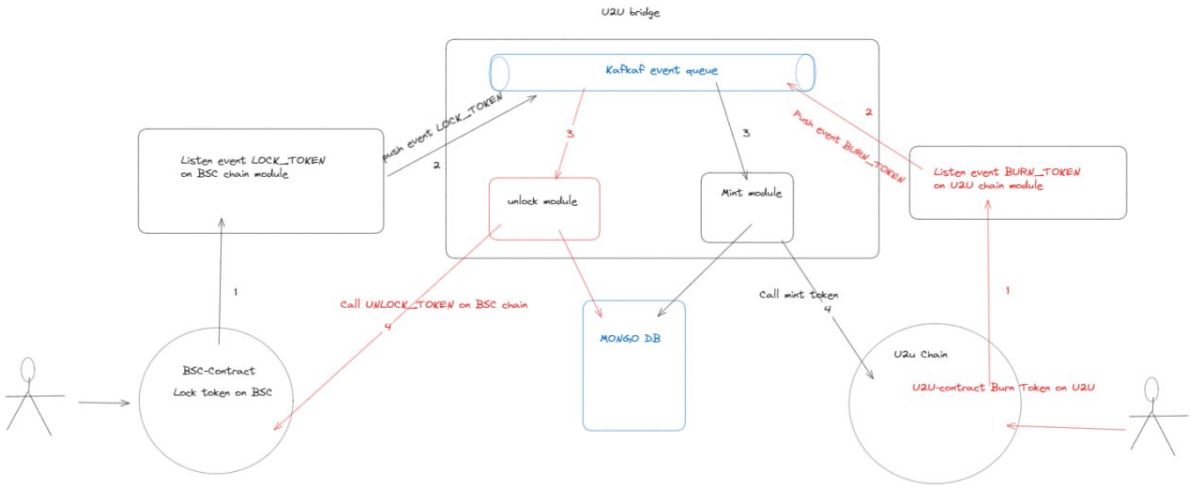

Image 1. Bridging process

1.2. Audit scope

This audit focused on identifying security flaws in code and the design of the U2U Bridge Smart Contract. It was conducted on commit `ddca168f4b41d140bf1bebe749c57b628761564b` from git repository link: *https://github.com/dstarter1/u2u-bridge-contract*.

1.3. Audit methodology

Our security audit process for smart contract includes two steps:

- Smart contract codes are scanned/tested for commonly known and more specific vulnerabilities using public and RK87, our in-house smart contract security analysis tool.
- 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 were considered during the audit of the smart contract:

- Integer Overflow and Underflow
- Timestamp Dependence

- Race Conditions
- Transaction-Ordering Dependence
- DoS with (Unexpected) revert
- DoS with Block Gas Limit
- Gas Usage, Gas Limit and Loops
- Redundant fallback function
- Unsafe type Inference
- Reentrancy
- Explicit visibility of functions state variables (external, internal, private and public)
- Logic Flaws

For vulnerabilities, we categorize the findings into categories as listed in table below, depending on their severity level:

SEVERITY LEVEL	DESCRIPTION
CRITICAL	A vulnerability that can disrupt the contract functioning; creates a critical risk to the contract; required to be fixed immediately.
HIGH	A vulnerability that could affect the desired outcome of executing the contract with high impact; needs to be fixed with high priority.
MEDIUM	A vulnerability that could affect the desired outcome of executing the contract with medium impact in a specific scenario; needs to be fixed.
LOW	An issue that does not have a significant impact, can be considered as less important.

Table 1. Severity levels

1.4. Disclaimer

Please note that security auditing cannot uncover all existing vulnerabilities, and even an audit in which no vulnerabilities are found is not a guarantee for a 100% secure smart contract. However, auditing allows discovering vulnerabilities that were unobserved, overlooked during development and areas where additional security measures are necessary.

2. AUDIT RESULT

2.1. Overview

The U2U Bridge Smart Contract was written in `Solidity` language, with the required version to be `^0.8.9`. The source code was written based on OpenZeppelin's library.

2.1.1. Tokens

- `BSCToken.sol` (U2V_BSC): is deployed on Binance Smart Chain, with total supply is 1_000_000*000x10*^{18} *(***Note: the number of decimals is 18, so the total representation token will be 1,000,000,000 or 1 billion.**_).
- `U2UToken.sol` (U2V_U2U): is deployed on U2U chain. The token enables the minter role to mint token.

2.1.2. `MultiSend.sol`

- The contract is deployed on Binance Smart Chain. It allows owner to set the amount of token allowed transfer from contract to another address.
- The owner can call `multiSend` function to execute a bridge token to U2U chain for a list of addresses supplied.

2.1.3. Bridge contract

- `U2UBridgeLock.sol`: is deployed on Binance Smart Chain.
- `U2UBridgeMint.sol`: is deployed on U2U chain

Threat modeling:

- Owner is able to:
  - turn on/off contract.
  - turn on/off EOA call.
  - turn on/off allow delay call.
  - set allow token.
  - set mod for per token.
  - set min amount to bridge.
- Mod per token is able to call function to complete bridge and mint/unlock token for receiver.
- Users are able to burn/lock token for bridging token.

2.2. Findings

During the audit process, the audit team found some vulnerabilities in the given version of U2U Bridge Smart Contract.

#	Severity	Name	Status
1	MEDIUM	Should have an emergency stop when meeting an incident	ACKNOWLEDGED
2	MEDIUM	Should use multi-signatures	ACKNOWLEDGED
3	INFORMATIVE	Missing checks for 0 address when setting state variables	FIXED

Table 2. Vulnerable issues list

2.2.1. [MEDIUM] Should have an emergency stop when meeting an incident

2.2.1.1. Description

If an illegal event occurs or vulnerabilities are discovered in your code, you must repair the defective code. Unfortunately, this is not possible while the code is executing. You must be able to implement an emergency stop that disables all calls to susceptible contract functions as part of your disaster recovery plan. These contract functions enable you to modify the code in question and resolve any difficulties that have emerged.

RECOMMENDATION

The contract may implement `Pausable.sol` from OpenZeppelin. Ref: *https://github.com/OpenZeppelin/openzeppelin-contracts/blob/master/contracts/utils/Pausable.sol*

UPDATES

- *Sep 18, 2023*: This issue has been acknowledged.

2.2.2. [MEDIUM] Should use multi-signatures

2.2.2.1. Description

In a bridge smart contract, multiple signature verification, often referred to as multi-signatures (multi-sigs) verification, is a security mechanism that requires multiple parties to sign off on a transaction or an action before it can be executed.

Multi-sigs add an element of security by ensuring that a single signer can't control the bridge. Multi-sigs might be used to enable the bridge contracts to be upgraded or paused. While

multi-sigs are an essential security control for bridges, they are not foolproof and require proper management

RECOMMENDATION

Once reach the multi-sig threshold, the transaction (unlock/mint) is executed.

UPDATES

- *Sep 18, 2023*: This issue has been acknowledged.

2.2.3. [INFORMATIVE] Missing checks for 0 address when setting states

Positions:

- U2UBridgeLock.sol#setTokenAllow()
- U2UBridgeLock.sol#setTokenMod()
- U2UBridgeLock.sol#setMinLockToken()
- U2UBridgeLock.sol#setOnlyEOACall()
- U2UBridgeMint.sol#setTokenAllow()
- U2UBridgeMint.sol#setTokenMod()
- U2UBridgeMint.sol#setMinBurnToken()
- U2UBridgeMint.sol#setOnlyEOACall()

2.2.3.1. Description

Checking addresses against zero-address during initialization or during setting is a security best-practice. However, such checks are missing in address variable initializations/changes in many places. Given that zero-address is used as an indicator for BNB, there is a greater risk of using it accidentally.

RECOMMENDATION

Add zero-address checks for setting relative address state variables.

UPDATES

- *Sep 18, 2023*: This issue has been acknowledged.
- *Sep 25, 2023*: This issue has been fixed.

3. VERSION HISTORY

Version	Date	Status/Change	Created by
1.0	*Sep 18, 2023*	Public Report	Verichains Lab
1.1	*Sep 25, 2023*	Public Report	Verichains Lab

Table 3. Report versions history