

# GOAT Network Audit Report

Tue Aug 27 2024



contact@bitslab.xyz



[https://twitter.com/scalebit\\_](https://twitter.com/scalebit_)



**ScaleBit**



# GOAT Network Audit Report

---

## 1 Executive Summary

### 1.1 Project Information

Description	GOAT Network is the first BTC L2 to share network ownership.
Type	Bridge
Auditors	ScaleBit
Timeline	Mon Aug 12 2024 - Thu Aug 22 2024
Languages	Solidity, Typescript
Platform	BTC
Methods	Architecture Review, Unit Testing, Manual Review
Source Code	<a href="https://github.com/GOATNetwork/goat-contracts">https://github.com/GOATNetwork/goat-contracts</a> <a href="https://github.com/GOATNetwork/btc-script-factory">https://github.com/GOATNetwork/btc-script-factory</a>
Commits	<a href="https://github.com/GOATNetwork/goat-contracts/commit/b8f3aa54e5423dba171966f74d6a032814c76d14d8e84088dd2a98a9d9216b0bc514fb7759b48d07">https://github.com/GOATNetwork/goat-contracts: b8f3aa54e5423dba171966f74d6a032814c76d14 d8e84088dd2a98a9d9216b0bc514fb7759b48d07</a> <a href="https://github.com/GOATNetwork/btc-script-factory/commit/44b96ae8fe6c2eb4d319ed897fa630c555b3e06926042a36f510eeecbe9d5e575fb54cfb8ce833e9">https://github.com/GOATNetwork/btc-script-factory: 44b96ae8fe6c2eb4d319ed897fa630c555b3e069 26042a36f510eeecbe9d5e575fb54cfb8ce833e9</a>

## 1.2 Files in Scope

The following are the SHA1 hashes of the original reviewed files.

ID	File	SHA-1 Hash
GTO	contracts/goat/GoatToken.sol	89bfa505fecda8acd15e27ea72163b3036bbd4cc
GFO	contracts/goat/GoatFoundation.sol	e098063e42c7464a53c67d91ce3737d96e28fa1f
BRI1	contracts/bridge/Bridge.sol	20041ca0bfe322eba94622147f2bb4d561edbcf2
LOC	contracts/locking/Locking.sol	c1c695878c0ec3293639b8fb0e13079e2c22a125
LSC	src/types/LockingScripts.ts	a405679337dc6a469119e471f0d6b2e66f9f49c2
UTXO	src/types/UTXO.ts	257af1f5d6c7c900e24969c0874bd0928eac890a
BSC	src/types/BridgeScripts.ts	19a40c19f7fb6c4939908fe0b79ce46daddfec65
TRA	src/types/transaction.ts	e485cd69f75b32decdbd98c7dd2746a1943b8fa3e
FEE	src/utils/fee.ts	381d3affe0e0fd1057b6f24d9eaa263f4c37245c
CUR	src/utils/curve.ts	b5d211bbd5bd2f778690e5891a8077fb23d1247b
BRI	src/covenantV1/bridge.ts	b8cc5c1554db0f2d6177163e87d0f91f27cf947f

LOC	src/covenantV1/locking.ts	83e28e6b1c9ea7b64e19246abf45 8cbabc2aef54
SCV1BST	src/covenantV1/bridge.script.ts	63ab140ce3706830effe0959d8d4c 793569f70fe
SCV1LST	src/covenantV1/locking.script.ts	4282ef9edfab6e5e9e2104d2737c5 73d88444c3c
SCR	src/slashable/bridge/script.ts	948f15b330152f17faa8edf78855be b2772926f9
SSBIT	src/slashable/bridge/index.ts	bf6fb571507418ceaaa281bb1dcbb 71ad6027366
SSLST	src/slashable/locking/script.ts	b44eec3baaee930c289125d0caed 8b8bea8fa095
SSLIT	src/slashable/locking/index.ts	f80f163b6296ff1ef0d5b21526eb44 62e141fbb2
SCIT	src/constants/index.ts	5a915d5e126c5d1335bb2bfeb7a1 eb2bd03041a1
IPU	src/constants/internalPubkey.ts	6014ea2646adbf2bd2efb917de4e d8228f20b901

## 1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	6	3	3
Informational	1	0	1
Minor	5	3	2
Medium	0	0	0
Major	0	0	0
Critical	0	0	0

## 1.4 ScaleBit Audit Breakdown

ScaleBit aims to assess repositories for security-related issues, code quality, and compliance with specifications and best practices. Possible issues our team looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Integer overflow/underflow
- Number of rounding errors
- Unchecked External Call
- Unchecked CALL Return Values
- Functionality Checks
- Reentrancy
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic issues
- Gas usage
- Fallback function usage
- tx.origin authentication
- Replay attacks
- Coding style issues

## 1.5 Methodology

The security team adopted the "**Testing and Automated Analysis**", "**Code Review**" and "**Formal Verification**" strategy to perform a complete security test on the code in a way that is closest to the real attack. The main entrance and scope of security testing are stated in the conventions in the "Audit Objective", which can expand to contexts beyond the scope according to the actual testing needs. The main types of this security audit include:

### (1) Testing and Automated Analysis

Items to check: state consistency / failure rollback / unit testing / value overflows / parameter verification / unhandled errors / boundary checking / coding specifications.

### (2) Code Review

The code scope is illustrated in section 1.2.

### (3) Audit Process

- Carry out relevant security tests on the testnet or the mainnet;
- If there are any questions during the audit process, communicate with the code owner in time. The code owners should actively cooperate (this might include providing the latest stable source code, relevant deployment scripts or methods, transaction signature scripts, exchange docking schemes, etc.);
- The necessary information during the audit process will be well documented for both the audit team and the code owner in a timely manner.

## 2 Summary

This report has been commissioned by [GOAT Network](#) to identify any potential issues and vulnerabilities in the source code of the [GOAT Network](#) smart contract, as well as any contract dependencies that were not part of an officially recognized library. In this audit, we have utilized various techniques, including manual code review and static analysis, to identify potential vulnerabilities and security issues.

During the audit, we identified 6 issues of varying severity, listed below.

ID	Title	Severity	Status
BRI-1	Transactions with No Change Will Result in an Additional Handling Fee	Minor	Fixed
BRI-2	User's UTXO May Contain Ordinals, Inscriptions, Or Other assets	Minor	Fixed
BRI1-1	Order of Amount Checks	Minor	Acknowledged
BRI1-2	Unused Function Decoration	Minor	Fixed
BRI1-3	Unused Variables	Minor	Acknowledged
BRI1-4	<code>_wid</code> Missing Check	Informational	Acknowledged



## 3 Participant Process

Here are the relevant actors with their respective abilities within the **GOAT Network** Smart Contract :

### Deployer

- The deployer will set the initial value of the contract in the `constructor` contract, including the data in `network` , `param` , `rateLimit` , `depositTaxBP` , `maxDepositTax` , `withdrawalTaxBP` , `maxWithdrawalTax` , `_res1` , `_res2` .

### User

- User can query `pubKeyHashAddrID` , `scriptHashAddrID` through the `base58Prefix` function.
- User can query the identifier through the `bech32HRP` function.
- User can query `name` through the `networkName` function.
- User can verify the address through the `isAddrValid` function.
- User can add a stake record to the target address and trigger a stake event through the `deposit` function.
- User can query whether `_txid` and the corresponding `_txout` have been used through the `isDeposited` function.
- User can initiate a withdrawal request through the `withdraw` function, and the initiator, recipient, and amount of funds are added to the queue, making the current request enter the `Pending` state.
- User can increase the `_maxTxPrice` in the corresponding `_wid` through the `replaceByFee` function.
- User can change his request in the `Pending` state to a request in the `Cancelling` state through the `cancel1` function.
- User can change the request in the `Canceled` state to a `Refunded` state through the `refund` function and receive a refund.

### OnlyRelayer

- Relayer can change the request in the `Pending` state or the `Cancelling` state to the `Canceled` state through the `cancel2` function.

- Relayer can change the request in the `Pending` state or the `Cancelling` state to the `Paid` state through the `paid` function, and send the `tax` fee to the `GoatFoundation`, and send the pledged funds to the fund black hole.

### **OnlyGoatFoundation**

- GoatFoundation can set the BP rate `depositTaxBP` during staking and the maximum `tax` limit `maxDepositTax` through the `setDepositTax` function.
- GoatFoundation can set the BP rate `withdrawalTaxBP` during withdrawal and the maximum `tax` limit `maxWithdrawalTax` through the `setDepositTax` function.
- GoatFoundation can set the buffer time `rateLimit` through the `setRateLimit` function.

## 4 Findings

### BRI-1 Transactions with No Change Will Result in an Additional Handling Fee

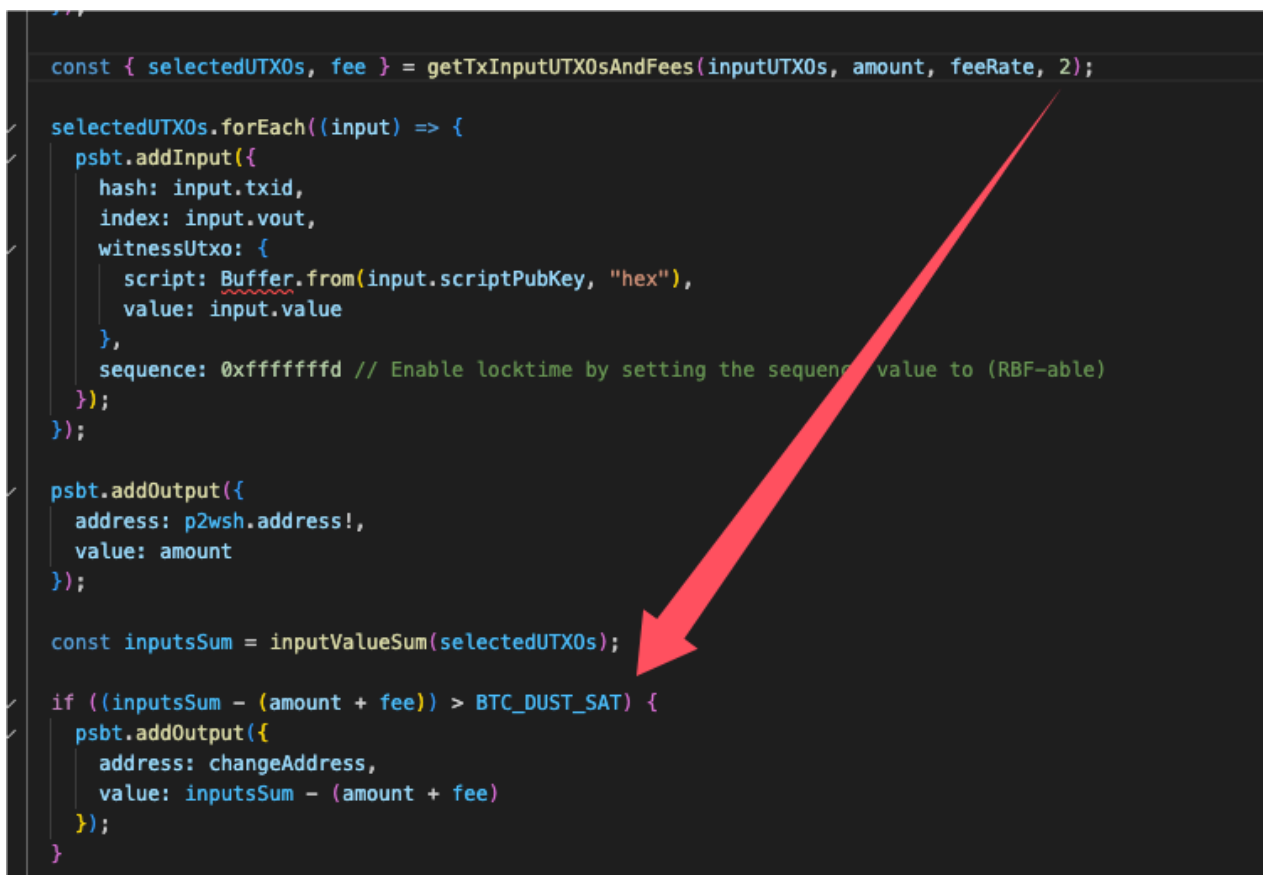
Severity: Minor

Status: Fixed

Code Location:

src/covenantV1/bridge.ts#39

Descriptions:



```
const { selectedUTXOs, fee } = getTxInputUTXOsAndFees(inputUTXOs, amount, feeRate, 2);

selectedUTXOs.forEach((input) => {
  psbt.addInput({
    hash: input.txid,
    index: input.vout,
    witnessUtxo: {
      script: Buffer.from(input.scriptPubKey, "hex"),
      value: input.value
    },
    sequence: 0xffffffff // Enable locktime by setting the sequence value to (RBF-able)
  });
});

psbt.addOutput({
  address: p2wsh.address!,
  value: amount
});

const inputsSum = inputValueSum(selectedUTXOs);

if ((inputsSum - (amount + fee)) > BTC_DUST_SAT) {
  psbt.addOutput({
    address: changeAddress,
    value: inputsSum - (amount + fee)
  });
}
```

When calculating the transaction fee, the user sets the output to two outputs by default, but if there is no change at the end, there is actually only one output, which is equivalent to the user paying one more output of the fee, resulting in an additional loss of funds.

Suggestion:

It is recommended that the handling fee be calculated on the basis of whether or not change is given.

# BRI-2 User's UTXO May Contain Ordinals, Inscriptions, Or Other assets

**Severity:** Minor

**Status:** Fixed

**Code Location:**

src/covenantV1/bridge.ts#18

**Descriptions:**

The user's UTXO may include assets such as Ordinals or inscriptions. It is recommended to filter out UTXOs containing these assets during the bridge process.

**Suggestion:**

Filter out UTXOs containing these assets during the bridge process.



# BRI1-1 Order of Amount Checks

**Severity:** Minor

**Status:** Acknowledged

**Code Location:**

contracts/bridge/Bridge.sol#193

**Descriptions:**

In the `withdraw` function, the check of `msg.value` is placed after the `tax` calculation.

When amount is less than the minimum limit, there is no need to calculate `tax` and `dust` .

**Suggestion:**

It is recommended to move require `amount > _maxTxPrice * constant` to the front.

## BRI1-2 Unused Function Decoration

**Severity:** Minor

**Status:** Fixed

**Code Location:**

contracts/bridge/Bridge.sol#219

**Descriptions:**

The Payable keyword is used in the function `replaceByFee`, but there is no logic processing related to `Msg.value` in the function. If the user uses this function to transfer funds to the contract, the amount in `withdrawals` will not be modified to lock the funds in the contract.

```
function replaceByFee(  
    uint256 _wid,  
    uint16 _maxTxPrice  
) external payable override {
```

**Suggestion:**

It is recommended to remove the `payable` keyword to ensure that funds are not locked.

## BRI1-3 Unused Variables

**Severity:** Minor

**Status:** Acknowledged

**Code Location:**

contracts/bridge/Bridge.sol#64,65

**Descriptions:**

In the `constructor` function, `_res1` and `_res2` have no other calls, and the structure defined in `Param` is not fully used. Is the initial value only used for testing? Is the initial value set in the `constructor` function in the contract.

**Suggestion:**

It is recommended to confirm the business logic and check.

## BRI1-4 `_wid` Missing Check

**Severity:** Informational

**Status:** Acknowledged

**Code Location:**

contracts/bridge/Bridge.sol#220

**Descriptions:**

In the external function `replaceByFee` , there is a lack of `_wid` value check. Any user call may access a non-existent index value, lacking checks and error prompts.

```
Withdrawal storage withdrawal = withdrawals[_wid];
```

Also in the `cancel1/refund` function.

**Suggestion:**

It is recommended to check if `_wid` is within the length range when reading `storage withdrawal` .



# Appendix 1

## Issue Level

- **Informational** issues are often recommendations to improve the style of the code or to optimize code that does not affect the overall functionality.
- **Minor** issues are general suggestions relevant to best practices and readability. They don't post any direct risk. Developers are encouraged to fix them.
- **Medium** issues are non-exploitable problems and not security vulnerabilities. They should be fixed unless there is a specific reason not to.
- **Major** issues are security vulnerabilities. They put a portion of users' sensitive information at risk, and often are not directly exploitable. All major issues should be fixed.
- **Critical** issues are directly exploitable security vulnerabilities. They put users' sensitive information at risk. All critical issues should be fixed.

## Issue Status

- **Fixed:** The issue has been resolved.
- **Partially Fixed:** The issue has been partially resolved.
- **Acknowledged:** The issue has been acknowledged by the code owner, and the code owner confirms it's as designed, and decides to keep it.

## Appendix 2

### Disclaimer

This report is based on the scope of materials and documents provided, with a limited review at the time provided. Results may not be complete and do not include all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your own risk. A report does not imply an endorsement of any particular project or team, nor does it guarantee its security. These reports should not be relied upon in any way by any third party, including for the purpose of making any decision to buy or sell products, services, or any other assets. TO THE FULLEST EXTENT PERMITTED BY LAW, WE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, IN CONNECTION WITH THIS REPORT, ITS CONTENT, RELATED SERVICES AND PRODUCTS, AND YOUR USE, INCLUDING BUT NOT LIMITED TO THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NOT INFRINGEMENT.

