GOAT Network
Audit Report

Tue Aug 27 2024

contact@bitslab.xyz 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	https://github.com/GOATNetwork/goat-contracts https://github.com/GOATNetwork/btc-script-factory
Commits	https://github.com/GOATNetwork/goat-contracts: b8f3aa54e5423dba171966f74d6a032814c76d14 d8e84088dd2a98a9d9216b0bc514fb7759b48d07 https://github.com/GOATNetwork/btc-script-factory: 44b96ae8fe6c2eb4d319ed897fa630c555b3e069 26042a36f510eeecbe9d5e575fb54cfb8ce833e9

1.2 Files in Scope

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

ID	File	SHA-1 Hash
MSS	protocol-units/settlement/mcr/contracts/src/staking/MovementStakingStorage.sol	9c44ab11e242531fb5661d437608a97f0970c177
IMS	protocol-units/settlement/mcr/contracts/src/staking/interfaces/IMovementStaking.sol	9a2b99b131c3069b228d4a680c485e7e5f723ac2
MST	protocol-units/settlement/mcr/contracts/src/staking/MovementStaking.sol	f1e10153092f27dc86c845af75ff0e551a6c7b24
BST	protocol-units/settlement/mcr/contracts/src/staking/base/BaseStaking.sol	fde371a66e7621ebd770a11976f29a6f47766b0c
MOVET	protocol-units/settlement/mcr/contracts/src/token/MOVEToken.sol	a1470c2184b93ed3c578713e72262d4e93cb9328
SMT	protocol-units/settlement/mcr/contracts/src/token/stlMoveToken.sol	2b29f08a07f667bfb98559c69ffa8b7de1a98fbc
CTO	protocol-units/settlement/mcr/contracts/src/token/custodian/CustodianToken.sol	95749fd646556f8b0148bc7aae47d1239d050546
LTO	protocol-units/settlement/mcr/contracts/src/token/locked/LockedToken.sol	7a9b686c7754c1128108e12df6a23233a74ef95e
LTS	protocol-units/settlement/mcr/contracts/src/token/locked/LockedTokenStorage.sol	24239047d457c08a8d50a140afc163958c956736

BTO	protocol-units/settlement/mcr/contracts/src/token/base/BaseToken.sol	90cea27dc655188098a759541917178a6214aad1
WTS	protocol-units/settlement/mcr/contracts/src/token/base/WrappedTokenStorage.sol	b301d6884e0e0c0f3870c22b2fa6458c14f7bc80
MTO	protocol-units/settlement/mcr/contracts/src/token/base/MintableToken.sol	dffd43d48f98c1804b964f7c35b148b2e4e81d45
WTO	protocol-units/settlement/mcr/contracts/src/token/base/WrappedToken.sol	dcc284d8fa4b2b0c7c8667268e75aec3fa1606ab
MCRS	protocol-units/settlement/mcr/contracts/src/settlement/MCRStorage.sol	fdf779ad0a6902bf2afd1b39fc07bcdb9498dda5
IMCR	protocol-units/settlement/mcr/contracts/src/settlement/interfaces/IMCR.sol	1ca49487ee52e81e4116b21d2307b0399ada2ba7
BSE	protocol-units/settlement/mcr/contracts/src/settlement/settlement/BaseSettlement.sol	e4d237cfc24c7c4011f292771260ff659b6e39a1
MCR	protocol-units/settlement/mcr/contracts/src/settlement/MCR.sol	c32e413f4c945e91338d58aa6c76e6b7fa53ecb4
MCRL	protocol-units/settlement/mcr/contracts/src/MCRLegacy.sol	85210a36c49d4a502b3ef290ba9069102cc6fec9

1.3 Issue Statistic

Item	Count	Fixed	Acknowledged
Total	7	0	0
Informational	2	0	0
Minor	1	0	0
Medium	0	0	0
Major	2	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 7 issues of varying severity, listed below.

ID	Title	Severity	Status
LTO-1	`Initialize` Could Be Front-Run	Minor	Pending
MCR-1	During The PoS Voting Process, `blockId` Can Be Arbitrarily Modified By An Attester, Bypassing The 2/3 Security Threshold	Informational	Pending
MST-1	`stake` Didn't Limit The Type of `custodian`	Major	Pending
MST-2	Inconsistency Between slash `amounts` and `refundamount`	Major	Pending
MST-3	`slash` Issue	Discussion	Pending
MST-4	`setGenesisCeremony` Issue	Discussion	Pending
MTO-1	Lack of Events Emit	Informational	Pending

3 Participant Process

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

111

4 Findings

LTO-1 `Initialize` Could Be Front-Run

Severity: Minor

Status: Pending

Code Location:

protocol-units/settlement/mcr/contracts/src/token/locked/LockedToken.sol#17;

protocol-units/settlement/mcr/contracts/src/token/custodian/CustodianToken.sol#45;

protocol-units/settlement/mcr/contracts/src/settlement/MCR.sol#13

Descriptions:

In the contract, by calling the `initialize` function to initialize the contracts, there is a potential issue that malicious attackers preemptively call the initialize function to initialize and there is no access control verification for the initialize functions.

Suggestion:

It is suggested that the `initialize` function can be called only by privileged addresses or in the same transaction immediately after the contract is created to avoid being maliciously called by the attacker.

MCR-1 During The PoS Voting Process, `blockId` Can Be Arbitrarily Modified By An Attester, Bypassing The 2/3 Security Threshold

Severity: Informational

Status: Pending

Code Location:

protocol-units/settlement/mcr/contracts/src/settlement/MCR.sol#186

Descriptions:

Below is the structure of `BlockCommitment`:

```
struct BlockCommitment {
    uint256 height;
    bytes32 commitment;
    bytes32 blockId;
}
```

During the voting process, the MCR contract only checks the `height` and `commitment` but does not check the `blockId`, allowing the first attester in the attesters list to arbitrarily modify the `blockId`.

Suggestion:

Add a check for `blockId` during the voting process.

MST-1 `stake` Didn't Limit The Type of `custodian`

Severity: Major

Status: Pending

Code Location:

protocol-units/settlement/mcr/contracts/src/staking/MovementStaking.sol#273

Descriptions:

Users with the `WHITELIST_ROLE` privilege can call the `stake` function to stake their `stake token`. However, there is no restriction on the type of `custodian` tokens, resulting in users being able to pass in worthless erc20 tokens and subsequently pay a small amount of `stake token` to the Staking contract making the if statements on `line 292` unequal, resulting in users being able to increase the number of stake at will.

Suggestion:

It is recommended to use whitelist mode for custodian tokens.

MST-2 Inconsistency Between slash `amounts` and `refundamount`

Severity: Major

Status: Pending

Code Location:

protocol-units/settlement/mcr/contracts/src/staking/MovementStaking.sol

Descriptions:

In the `unstake` and `stake` functions, the number of stakes added is the same as the number of `custodians` transferred in. In the `slash` function the `refundAmount` is the min of the stake balance, the amount to be slashed, and the refund amount, but the number of stakes to slash the user is using the function parameter `amounts`.

Suggestion:

It is recommended to ensure that this is as designed and fixed to the correct amount.

MST-3 `slash` Issue

Severity: Discussion

Status: Pending

Code Location:

protocol-units/settlement/mcr/contracts/src/staking/MovementStaking.sol

Descriptions:

The slash mechanism makes the nodes accountable for their actions and can deter malicious attack behavior and negligence, ensuring a robust and trustworthy network infrastructure. But the slash function in the contract can only manipulate data from `msg.sender`, meaning it can only be called by the user who is being slashed himself, and all the parameters are controllable, it's needed to make sure that this is by design.

Suggestion:

It is recommended to ensure that this is as designed.

MST-4 `setGenesisCeremony` Issue

Severity: Discussion

Status: Pending

Code Location:

protocol-units/settlement/mcr/contracts/src/staking/MovementStaking.sol#93

Descriptions:

Since the `refundAmount` paid by `setGenesisCeremony` to `attesters` is derived from the `MovementStaking` contract, it is possible for users to register new `domains` in order to withdraw tokens.

Suggestion:

It is recommended to ensure that this is as designed.

MTO-1 Lack of Events Emit

Severity: Informational

Status: Pending

Code Location:

protocol-units/settlement/mcr/contracts/src/token/base/MintableToken.sol#51

Descriptions:

The contract lacks appropriate events for monitoring `grantMinterRole()` operations, which could make it difficult to track sensitive actions or detect potential issues.

Suggestion:

It is recommended to emit events for the those function.

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