MixBytes()

Mantle Network fBTC Timelock Security Audit Report

MAY 28, 2025

Table of Contents

1. Introduction

1.1 Disclaimer

The audit makes no statements or warranties regarding the utility, safety, or security of the code, the suitability of the business model, investment advice, endorsement of the platform or its products, the regulatory regime for the business model, or any other claims about the fitness of the contracts for a particular purpose or their bug-free status.

1.2 Executive Summary

TimelockSafeGuard is a contract that transforms a Gnosis Safe multisig into a timelock contract. It allows scheduling transactions with execution delays.

The audit was conducted over 1 day by 3 auditors, involving a thorough manual code review of the scope and analysis via automated tools.

During the audit, the following potential attack vectors were analyzed:

1. **Reentrancy**:
   We examined that the contract contains no external calls before state changes.

2. **Underflow**:
   We verified that _operationCount[id] is always greater than zero before being decremented.

3. **Replay After Cancellation**:
   We reviewed that once an operation is canceled, its execution count cannot be reused until it is scheduled again.

4. **Timelock Bypass**:
   We checked whether setting minDelay to zero allows proposals to be executed without any delay.

5. **Authorization Inconsistency**:
   We evaluated that role-checking behavior varies across different scheduling entry points, leading to inconsistent authorization patterns.

6. **Configuration Consistency**:
   We inspected that changing minDelay after an operation is scheduled does not alter the required delay for that pending operation.

7. **Gnosis Safe Integration**:
   We analyzed the checkTransaction and checkAfterExecution hooks to see how they integrate with Gnosis Safe's execution workflow.

8. **Proposal Deadlines**:
   We checked for the presence of an expiration mechanism for scheduled proposals.

9. **Proposal Consistency and Parameter Mutability**:
   We checked that once a proposal is scheduled, its parameters cannot be changed.

We also recommend paying close attention to the accounts granted various roles within the project, as misconfiguration or misuse can lead to unexpected consequences.

1.3 Project Overview

Summary

Title	Description
Client Name	Mantle Network
Project Name	fBTC Timelock
Type	Solidity
Platform	EVM
Timeline	13.05.2025 - 27.05.2025

Scope of Audit

File	Link
src/TimeLockSafeGuard.sol	TimeLockSafeGuard.sol

Versions Log

Date	Commit Hash	Note
13.05.2025	6d54c09515dc7dda6346927fb6b0de84f77a5ef4	Initial Commit
26.05.2025	f84c6c92d1a1ba82edb458b6c1e62896d51f4474	Re-audit commit

Mainnet Deployments

File	Address	Blockchain

File	Address	Blockchain
TimeLockSafeGuard.sol	0xe15F7F...87fA5bA4	Ethereum
TimeLockSafeGuard.sol	0xb56D62...F3868f8b	Mantle

1.4 Security Assessment Methodology

Project Flow

Stage	Scope of Work
Interim audit	**Project Architecture Review:** · Review project documentation · Conduct a general code review · Perform reverse engineering to analyze the project's architecture based solely on the source code · Develop an independent perspective on the project's architecture · Identify any logical flaws in the design OBJECTIVE: UNDERSTAND THE OVERALL STRUCTURE OF THE PROJECT AND IDENTIFY POTENTIAL SECURITY RISKS.
	Code Review with a Hacker Mindset: · Each team member independently conducts a manual code review, focusing on identifying unique vulnerabilities. · Perform collaborative audits (pair auditing) of the most complex code sections, supervised by the Team Lead. · Develop Proof-of-Concepts (PoCs) and conduct fuzzing tests using tools like Foundry, Hardhat, and BOA to uncover intricate logical flaws. · Review test cases and in-code comments to identify potential weaknesses. OBJECTIVE: IDENTIFY AND ELIMINATE THE MAJORITY OF VULNERABILITIES, INCLUDING THOSE UNIQUE TO THE INDUSTRY.
	Code Review with a Nerd Mindset: · Conduct a manual code review using an internally maintained checklist, regularly updated with insights from past hacks, research, and client audits. · Utilize static analysis tools (e.g., Slither, Mythril) and vulnerability databases (e.g., Solodit) to uncover potential undetected attack vectors. OBJECTIVE: ENSURE COMPREHENSIVE COVERAGE OF ALL KNOWN ATTACK VECTORS DURING THE REVIEW PROCESS.

Stage	Scope of Work
	Consolidation of Auditors' Reports: · Cross-check findings among auditors · Discuss identified issues · Issue an interim audit report for client review OBJECTIVE: COMBINE INTERIM REPORTS FROM ALL AUDITORS INTO A SINGLE COMPREHENSIVE DOCUMENT.
Re-audit	**Bug Fixing & Re-Audit:** · The client addresses the identified issues and provides feedback · Auditors verify the fixes and update their statuses with supporting evidence · A re-audit report is generated and shared with the client OBJECTIVE: VALIDATE THE FIXES AND REASSESS THE CODE TO ENSURE ALL VULNERABILITIES ARE RESOLVED AND NO NEW VULNERABILITIES ARE ADDED.
Final audit	**Final Code Verification & Public Audit Report:** · Verify the final code version against recommendations and their statuses · Check deployed contracts for correct initialization parameters · Confirm that the deployed code matches the audited version · Issue a public audit report, published on our official GitHub repository · Announce the successful audit on our official X account OBJECTIVE: PERFORM A FINAL REVIEW AND ISSUE A PUBLIC REPORT DOCUMENTING THE AUDIT.

1.5 Risk Classification

Severity Level Matrix

Severity	Impact: High	Impact: Medium	Impact: Low
Likehood: High	Critical	High	Medium
Likehood: Medium	High	Medium	Low
Likehood: Low	Medium	Low	Low

Impact

- **High** – Theft from 0.5% OR partial/full blocking of funds (>0.5%) on the contract without the possibility of withdrawal OR loss of user funds (>1%) who interacted with the protocol.
- **Medium** – Contract lock that can only be fixed through a contract upgrade OR one-time theft of rewards or an amount up to 0.5% of the protocol's TVL OR funds lock with the possibility of withdrawal by an admin.
- **Low** – One-time contract lock that can be fixed by the administrator without a contract upgrade.

Likelihood

- **High** – The event has a 50–60% probability of occurring within a year and can be triggered by any actor (e.g., due to a likely market condition that the actor cannot influence).
- **Medium** – An unlikely event (10–20% probability of occurring) that can be triggered by a trusted actor.
- **Low** – A highly unlikely event that can only be triggered by the owner.

Action Required

- **Critical** – Must be fixed as soon as possible.
- **High** – Strongly advised to be fixed to minimize potential risks.
- **Medium** – Recommended to be fixed to enhance security and stability.
- **Low** – Recommended to be fixed to improve overall robustness and effectiveness.

Finding Status

- **Fixed** – The recommended fixes have been implemented in the project code and no longer impact its security.
- **Partially Fixed** – The recommended fixes have been partially implemented, reducing the impact of the finding, but it has not been fully resolved.
- **Acknowledged** – The recommended fixes have not yet been implemented, and the finding remains unresolved or does not require code changes.

1.6 Summary of Findings

Findings Count

Severity	Count
Critical	0
High	0
Medium	3
Low	8

Findings Statuses

ID	Finding	Severity	Status
M-1	`TimelockSafeGuard.cancel` Not Whitelisted for Immediate Execution	Medium	Fixed
M-2	Malicious Canceller Can Lock the Safe	Medium	Fixed
M-3	`TimelockSafeGuard.updateMinDelay` Lacks Minimum and Maximum Limits	Medium	Fixed
L-1	Single Admin Can Freeze the Safe	Low	Fixed
L-2	Gas Price Inside Operation Hash Can Delay Execution	Low	Acknowledged
L-3	`TimelockSafeGuard.hashOperation` Marked `view` Instead of `pure`	Low	Fixed
L-4	Revocable `SAFE_ROLE` Can Lock the Safe Wallet	Low	Fixed
L-5	Inconsistent role checking between `schedule` and `scheduleSimple` functions	Low	Fixed
L-6	Operations with zero count can be scheduled but never executed	Low	Fixed
L-7	Inconsistent Open Role Logic for `EXECUTOR` and `PROPOSER` Roles	Low	Acknowledged

L-8	Immutable Execution Count for Scheduled Operations	Low	Acknowledged

2. Findings Report

2.1 Critical

Not Found

2.2 High

Not Found

2.3 Medium

M-1	TimelockSafeGuard.cancel Not Whitelisted for Immediate Execution		
Severity	Medium	**Status**	Fixed in f84c6c92

Description

TimelockSafeGuard.checkTransaction skips the timelock for schedule and scheduleSimple but still enforces the delay for cancel. If the Safe itself is a CANCELLER_ROLE holder, any attempt to cancel a malicious proposal must wait out the full delay. A compromised proposer can queue a harmful transaction, leaving owners without an instant remedy unless another address with CANCELLER_ROLE exists outside the guard. Because timely cancellation is an essential defensive action, withholding it creates a meaningful attack window and warrants **Medium** severity.

Recommendation

We recommend allowing authorised cancellers to call cancel without delay.

Client's Commentary:

Fixed. f84c6c92

M-2	Malicious Canceller Can Lock the Safe		
Severity	Medium	**Status**	Fixed in f84c6c92

Description

An address with CANCELLER_ROLE can veto any queued proposal. If that address becomes compromised while DEFAULT_ADMIN_ROLE is held by the Safe, revoking the role requires a timelocked transaction that the attacker can cancel repeatedly. The Safe becomes permanently unable to execute any operation, including the role-revocation and disabling the guard contract, if they were not whitelisted preemptively. This places overall control at risk, giving the issue **Medium** severity.

Recommendation

We recommend allowing an authorised admin to call revokeRole without delay.

Client's Commentary:

Fixed. please refer to f84c6c92

M-3	TimelockSafeGuard.updateMinDelay Lacks Minimum and Maximum Limits		
Severity	Medium	Status	Fixed in f84c6c92

Description

The TimelockSafeGuard.updateMinDelay function allows the DEFAULT_ADMIN_ROLE to set any value for the **_minDelay**, including an unreasonably large value that could effectively freeze all future proposals. Proposals with excessively long delays might still be executed far in the future, which may be unexpected. Additionally, the **_minDelay** can be set to zero, enabling immediate execution of proposals and effectively disabling the timelock feature.

Recommendation

We recommend adding both minimum and maximum bounds to the **minDelay** variable to impose a reasonable upper limit and prevent it from being set to zero.

Client's Commentary:

Client: Add "30 days" as the maximum limit. Not necessary to set minimum limit. f84c6c92

MixBytes: The current implementation of `TimelockSafeGuard._schedule` allows setting a very large delay value, which may result in `_timestamps[id]` being set far in the future. We also recommend adding an upper bound check on `delay` to prevent excessively large values and to avoid unexpected execution of long-scheduled proposals.

Client: We are aware of this, and will keep the current implementation and keep the possibility to set a delay > 30 days. If a proposal was configured with a excessively large values, we cancel it with no delay.

2.4 Low

L-1	Single Admin Can Freeze the Safe		
Severity	Low	**Status**	Fixed in f84c6c92

Description

The holder of DEFAULT_ADMIN_ROLE can revoke every other role or raise _minDelay to an extreme value. Any of these actions stops new transactions or blocks those already queued. Centralising this power in one key creates a single-point-of-failure denial-of-service risk. Impact is chiefly on availability, so the finding is **Low** severity.

Recommendation

We recommend assigning DEFAULT_ADMIN_ROLE only to a Safe itself, multisig or DAO with a stronger or equal trust model than a Safe address. Alternatively add an immutable mechanism for owners to disconnect a malfunctioning guard.

Client's Commentary:

Fixed. We replace onlyRole(DEFAULT_ADMIN_ROLE) with onlySafe. f84c6c92

L-2	Gas Price Inside Operation Hash Can Delay Execution		
Severity	Low	**Status**	Acknowledged

Description

`TimelockSafeGuard.hashOperation` function includes `gasPrice` parameter, but wallets typically set an up-to-date gas price at execution. If the live value differs from the scheduled one, `TimelockSafeGuard.checkTransaction` computes a non-matching hash and rejects the call. Owners must replay the transaction with the stale gas price or abandon it, potentially stalling the Safe beyond the intended delay.

Recommendation

Remove `gasPrice` from the hash pre-image, or permit an execution-time override within a bounded range.

Client's Commentary:

gasPrice will be not used, and will always keep as 0.

L-3	TimelockSafeGuard.hashOperation Marked view Instead of pure		
Severity	Low	**Status**	Fixed in f84c6c92

Description

hashOperation reads no state but is declared view. Using pure better reflects its behaviour, enables compiler optimisations, and aids static analysis.

Recommendation

Change the function declaration from public view to public pure.

Client's Commentary:

Fixed.

L-4	Revocable SAFE_ROLE Can Lock the Safe Wallet		
Severity	Low	**Status**	Fixed in f84c6c92

Description

The SAFE_ROLE, that defines the Safe contract, is granted and revocable like any other role. A malicious or compromised admin could revoke it, after which every Safe transaction fails the guard's role check, permanently disabling the wallet. Because exploitation requires admin action, the severity is **Low**, but the impact is total loss of functionality.

Recommendation

We recommend making the Safe address immutable in constructor.

Client's Commentary:

Fixed it. We delete the SAFE_ROLE, and use onlySafe.

L-5	Inconsistent role checking between `schedule` and `scheduleSimple` functions		
Severity	Low	**Status**	Fixed in `f84c6c92`

Description

There is an inconsistency in role checking between the `TimelockSafeGuard.schedule` and `TimelockSafeGuard.scheduleSimple` functions. While `schedule` uses `TimelockSafeGuard._checkRoleOrOpenRole`, which allows anyone to call the function if the role is open, `scheduleSimple` uses `onlyRole`, which strictly requires the caller to have the role. This creates unexpected behavior where, even if the proposer role is open, users can only use `schedule` but not `scheduleSimple`.

Recommendation

We recommend removing the redundant `onlyRole(PROPOSER_ROLE)` modifier from `scheduleSimple` since it already uses the same role check through the schedule function call.

Client's Commentary:

Fixed it, delete onlyRole(PROPOSER_ROLE) modifier for scheduleSimple. f84c6c92

L-6	Operations with zero count can be scheduled but never executed		
Severity	Low	**Status**	Fixed in f84c6c92

Description

If `count` parameter is set to zero in the `TimelockSafeGuard.schedule`, the operation will be scheduled but can never be executed because `TimelockSafeGuard.isOperationReady` checks for `getOperationCount(id) > 0`, effectively creating an inactive operation that can only be cancelled.

Recommendation

We recommend adding a require statement in the `TimelockSafeGuard.schedule` function to ensure that `count` is greater than zero:

```
require(count > 0, "TimelockSafeGuard: count must be > 0");
```

Client's Commentary:

Fixed. f84c6c92

L-7	Inconsistent Open Role Logic for EXECUTOR and PROPOSER Roles		
Severity	Low	**Status**	Acknowledged

Description

The open roles mechanism for the EXECUTOR and PROPOSER roles is implemented by granting the role to address(0), allowing any account to pass the check. However, this behaves inconsistently when role memberships change:

1. Empty Role Membership ≠ Open Role
   If all explicit members of a role are removed (so the set becomes empty), the contract does not automatically grant the open-role via address(0). As a result, no account — not even originally granted ones — can pass the role check.
2. Adding Members to an Open Role Does Not Close It
   Conversely, if a role was previously open and then a real account is granted that role, the open - role flag remains active. The role never closes and any address continues to pass the check alongside the newly granted member.

This inconsistency can lead to unexpected authorization failures (when roles become accidentally empty) or unintended open access (when roles are thought to be restricted but remain open

Recommendation

Redesign the access - control logic to separate the concepts of open and member roles explicitly.

Client's Commentary:

We are aware of this, and will keep the current implementation.

L-8	Immutable Execution Count for Scheduled Operations		
Severity	Low	**Status**	Acknowledged

Description

Each scheduled proposal is initialized with a count parameter that determines how many times it may be executed. Once a proposal is created, you must either wait for it to decrement to zero through executions or cancel and recreate the proposal to adjust its execution allowance.

Recommendation

Provide administrative functions to adjust the execution count of a pending operation without full cancellation.

Client's Commentary:

We are aware of this and will keep current implmentation.

3. About MixBytes

MixBytes is a leading provider of smart contract audit and research services, helping blockchain projects enhance security and reliability. Since its inception, MixBytes has been committed to safeguarding the Web3 ecosystem by delivering rigorous security assessments and cutting-edge research tailored to DeFi projects.

Our team comprises highly skilled engineers, security experts, and blockchain researchers with deep expertise in formal verification, smart contract auditing, and protocol research. With proven experience in Web3, MixBytes combines in-depth technical knowledge with a proactive security-first approach.

Why MixBytes

- **Proven Track Record:** Trusted by top-tier blockchain projects like Lido, Aave, Curve, and others, MixBytes has successfully audited and secured billions in digital assets.
- **Technical Expertise:** Our auditors and researchers hold advanced degrees in cryptography, cybersecurity, and distributed systems.
- **Innovative Research:** Our team actively contributes to blockchain security research, sharing knowledge with the community.

Our Services

- **Smart Contract Audits:** A meticulous security assessment of DeFi protocols to prevent vulnerabilities before deployment.
- **Blockchain Research:** In-depth technical research and security modeling for Web3 projects.
- **Custom Security Solutions:** Tailored security frameworks for complex decentralized applications and blockchain ecosystems.

MixBytes is dedicated to securing the future of blockchain technology by delivering unparalleled security expertise and research-driven solutions. Whether you are launching a DeFi protocol or developing an innovative dApp, we are your trusted security partner.

Contact Information

https://mixbytes.io/

https://github.com/mixbytes/audits_public

hello@mixbytes.io

https://x.com/mixbytes