Audited by CRED SHIELDS

CredShields

Smart Contract Audit

May 7, 2026 • CONFIDENTIAL

Description

This document details the process and result of the Smart Contract audit performed by CredShields Technologies PTE. LTD. on behalf of Austech Solutions LTD between April 20, 2026, and April 21, 2026. A retest was performed on April 28, 2026.

Author

Shashank (Co-founder, CredShields) shashank@CredShields.com

Reviewers

Aditya Dixit (Research Team Lead), Shreyas Koli (Auditor), Sanket Salavi (Auditor), Prasad Kuri (Auditor), Neel Shah (Auditor)

Prepared for

Austech Solutions LTD

AICPA SOC

SCS Advocates

OWASP Smart Contract Top 10 Security Risks & Vulnerabilities

Table of Contents

Table of Contents **2**
1. Executive Summary ——————————- **3**
State of Security 4
2. The Methodology ——————————- **5**
2.1 Preparation Phase 5
2.1.1 Scope 5
2.1.2 Documentation 5
2.1.3 Audit Goals 6
2.2 Retesting Phase 6
2.3 Vulnerability classification and severity 6
2.4 CredShields staff 8
3. Findings Summary ——————————- **9**
3.1 Findings Overview 9
3.1.1 Vulnerability Summary 9
5. Bug Reports —————————— **10**
Bug ID #L001 [Acknowledged] 10
Owner Can Mint Unlimited Tokens Without Supply Cap 10
Bug ID #I001 [Acknowledged] 11
Use Ownable2Step 11
Bug ID #I002 [Acknowledged] 12
Hardcoded Static Address 12
Bug ID #G001 [Acknowledged] 13
Public constants can be private 13
6. The Disclosure ——————————- **14**

1. Executive Summary ----------

Austech Solutions LTD engaged CredShields to perform a smart contract audit from April 20, 2026, to April 21, 2026. During this timeframe, 4 vulnerabilities were identified. **A retest was performed on April 28, 2026, and all the bugs have been addressed.**

During the audit, 0 vulnerabilities were found with a severity rating of either High or Critical. These vulnerabilities represent the greatest immediate risk to "Austech Solutions LTD" and should be prioritized for remediation, and fortunately, none were found.

The table below shows the in-scope assets and a breakdown of findings by severity per asset. Section 2.3 contains more information on how severity is calculated.

Assets in Scope	Critical	High	Medium	Low	info	Gas	Σ
Manhattan Token	0	0	0	1	2	1	**4**

Table: Vulnerabilities Per Asset in Scope

The CredShields team conducted the security audit to focus on identifying vulnerabilities in Manhattan Token's scope during the testing window while abiding by the policies set forth by Austech Solutions LTD's team.

State of Security

To maintain a robust security posture, it is essential to continuously review and improve upon current security processes. Utilizing CredShields' continuous audit feature allows both Austech Solutions LTD's internal security and development teams to not only identify specific vulnerabilities but also gain a deeper understanding of the current security threat landscape.

To ensure that vulnerabilities are not introduced when new features are added, or code is refactored, we recommend conducting regular security assessments. Additionally, by analyzing the root cause of resolved vulnerabilities, the internal teams at Austech Solutions LTD can implement both manual and automated procedures to eliminate entire classes of vulnerabilities in the future. By taking a proactive approach, Austech Solutions LTD can future-proof its security posture and protect its assets.

2. The Methodology ----------

Austech Solutions LTD engaged CredShields to perform a Manhattan Token audit. The following sections cover how the engagement was put together and executed.

2.1 Preparation Phase

The CredShields team meticulously reviewed all provided documents and comments in the smart contract code to gain a thorough understanding of the contract's features and functionalities. They meticulously examined all functions and created a mind map to systematically identify potential security vulnerabilities, prioritizing those that were more critical and business-sensitive for the refactored code. To confirm their findings, the team deployed a self-hosted version of the smart contract and performed verifications and validations during the audit phase.

A testing window from April 20, 2026, to April 21, 2026, was agreed upon during the preparation phase.

2.1.1 Scope

During the preparation phase, the following scope for the engagement was agreed upon:

IN SCOPE ASSETS
Audited Scope: https://etherscan.io/token/0x8c881e37b357B74F2C42a1Ae5d1c908078753923#code

2.1.2 Documentation

Documentation was not required as the code was self-sufficient for understanding the project.

2.1.3 Audit Goals

CredShields employs a combination of in-house tools and thorough manual review processes to deliver comprehensive smart contract security audits. The majority of the audit involves manual inspection of the contract's source code, guided by OWASP's Smart Contract Security Weakness Enumeration (SCWE) framework and an extended, self-developed checklist built from industry best practices. The team focuses on deeply understanding the contract's core logic, designing targeted test cases, and assessing business logic for potential vulnerabilities across OWASP's identified weakness classes.

CredShields aligns its auditing methodology with the OWASP Smart Contract Security projects, including the Smart Contract Security Verification Standard (SCSVS), the Smart Contract Weakness Enumeration (SCWE), and the Smart Contract Secure Testing Guide (SCSTG). These frameworks, actively contributed to and co-developed by the CredShields team, aim to bring consistency, clarity, and depth to smart contract security assessments. By adhering to these OWASP standards, we ensure that each audit is performed against a transparent, community-driven, and technically robust baseline. This approach enables us to deliver structured, high-quality audits that address both common and complex smart contract vulnerabilities systematically.

2.2 Retesting Phase

Austech Solutions LTD is actively partnering with CredShields to validate the remediations implemented towards the discovered vulnerabilities.

2.3 Vulnerability classification and severity

CredShields follows OWASP's Risk Rating Methodology to determine the risk associated with discovered vulnerabilities. This approach considers two factors - Likelihood and Impact - which are evaluated with three possible values - **Low**, **Medium**, and **High**, based on factors such as Threat

agents, Vulnerability factors, and Technical and Business Impacts. The overall severity of the risk is calculated by combining the likelihood and impact estimates.

Overall Risk Severity				
Impact	HIGH	Medium	High	Critical
	MEDIUM	Low	Medium	High
	LOW	None	Low	Medium
		LOW	MEDIUM	HIGH
Likelihood				

Overall, the categories can be defined as described below -

1. **Informational**

   We prioritize technical excellence and pay attention to detail in our coding practices. Our guidelines, standards, and best practices help ensure software stability and reliability. Informational vulnerabilities are opportunities for improvement and do not pose a direct risk to the contract. Code maintainers should use their own judgment on whether to address them.

2. **Low**

   Low-risk vulnerabilities are those that either have a small impact or can't be exploited repeatedly or those the client considers insignificant based on their specific business circumstances.

3. **Medium**

   Medium-severity vulnerabilities are those caused by weak or flawed logic in the code and can lead to exfiltration or modification of private user information. These vulnerabilities

can harm the client's reputation under certain conditions and should be fixed within a specified timeframe.

4. **High**

   High-severity vulnerabilities pose a significant risk to the Smart Contract and the organization. They can result in the loss of funds for some users, may or may not require specific conditions, and are more complex to exploit. These vulnerabilities can harm the client's reputation and should be fixed immediately.

5. **Critical**

   Critical issues are directly exploitable bugs or security vulnerabilities that do not require specific conditions. They often result in the loss of funds and Ether from Smart Contracts or users and put sensitive user information at risk of compromise or modification. The client's reputation and financial stability will be severely impacted if these issues are not addressed immediately.

6. **Gas**

   To address the risk and volatility of smart contracts and the use of gas as a method of payment, CredShields has introduced a "Gas" severity category. This category deals with optimizing code and refactoring to conserve gas.

2.4 CredShields staff

The following individual at CredShields managed this engagement and produced this report:

- Shashank, Co-founder CredShields shashank@CredShields.com

Please feel free to contact this individual with any questions or concerns you have about the engagement or this document.

3. Findings Summary ----------

This chapter contains the results of the security assessment. Findings are sorted by their severity and grouped by asset and OWASP SCWE classification. Each asset section includes a summary highlighting the key risks and observations. The table in the executive summary presents the total number of identified security vulnerabilities per asset, categorized by risk severity based on the OWASP Smart Contract Security Weakness Enumeration framework.

3.1 Findings Overview

3.1.1 Vulnerability Summary

During the security assessment, 4 security vulnerabilities were identified in the asset.

Vulnerability Title	Severity	Vulnerability Type	Status
Owner Can Mint Unlimited Tokens Without Supply Cap	Low	Access Control (SCWE-016)	Acknowledged
Use Ownable2Step	Informational	Missing Best Practices (SCWE-008)	Acknowledged
Hardcoded Static Address	Informational	Missing Best Practices (SCWE-008)	Acknowledged
Public constants can be private	Gas	Gas Optimization (SCWE-082)	Acknowledged

Table: Findings and Remediations

5. Bug Reports ----------

Bug ID #L001 [Acknowledged]

Owner Can Mint Unlimited Tokens Without Supply Cap

Vulnerability Type
Access Control (SCWE-016)

Severity
Low

Description
The ManhattanToken contract declares five reserve constants (presaleReserve, loyaltyRewardsAndStakingReserve, treasuryAndLiquidityReserve, affiliatesReserve, reserves) that sum to exactly 1,000,000,000 MPCN. However, the contract also exposes an onlyOwner mint(address,uint256) function with no per-call limit, no global supply cap, no time lock, and no rate limit, allowing the owner to create arbitrary additional tokens at any time and silently break the fixed-supply invariant that the reserve constants imply. This introduces a significant centralization risk, as mint authority is concentrated in a single EOA using single-step Ownable, meaning a compromise, loss, or accidental transfer of that key directly translates into either unbounded inflation or a permanently stranded mint privilege.

Affected Code

- 0x8c881e37b357B74F2C42a1Ae5d1c908078753923#code#F1#L24

Impacts
The owner can unilaterally mint any amount of MPCN at any moment, diluting every existing holder.

Remediation
Enforce a hard cap, so that _mint automatically reverts once the cap is reached.

Retest
Client's Comments: The client acknowledged the uncapped minting finding, stating it is a functional requirement for their membership utility token to ensure supply meets network demand without the restrictions of a speculative secondary market.

Bug ID #I001 [Acknowledged]

Use Ownable2Step

Vulnerability Type
Missing Best Practices

Severity
Informational

Description
The "Ownable2Step" pattern is an improvement over the traditional "Ownable" pattern, designed to enhance the security of ownership transfer functionality in a smart contract. Unlike the original "Ownable" pattern, where ownership can be transferred directly to a specified address, the "Ownable2Step" pattern introduces an additional step in the ownership transfer process. Ownership transfer only completes when the proposed new owner explicitly accepts the ownership, mitigating the risk of accidental or unintended ownership transfers to mistyped addresses.

Affected Code

- 0x8c881e37b357B74F2C42a1Ae5d1c908078753923#code#F1#L9

Impacts
Without the "Ownable2Step" pattern, the contract owner might inadvertently transfer ownership to an unintended or mistyped address, potentially leading to a loss of control over the contract. By adopting the "Ownable2Step" pattern, the smart contract becomes more resilient against external attacks aimed at seizing ownership or manipulating the contract's behavior.

Remediation
It is recommended to use either Ownable2Step or Ownable2StepUpgradeable depending on the smart contract.

Retest:
Client Comment: Acknowledged

Bug ID #I002 [Acknowledged]

Hardcoded Static Address

Vulnerability Type
Missing Best Practices (SCWE-008)

Severity
Informational

Description
The contract was found to be using hardcoded addresses.
This could have been optimized using dynamic address update techniques along with proper access control to aid in address upgrade at a later stage.

Affected Code

- 0x8c881e37b357B74F2C42a1Ae5d1c908078753923#code#F1#L17
- 0x8c881e37b357B74F2C42a1Ae5d1c908078753923#code#F1#L18
- 0x8c881e37b357B74F2C42a1Ae5d1c908078753923#code#F1#L19
- 0x8c881e37b357B74F2C42a1Ae5d1c908078753923#code#F1#L20
- 0x8c881e37b357B74F2C42a1Ae5d1c908078753923#code#F1#L21

Impacts
Hardcoding address variables in the contract make it difficult for it to be modified at a later stage in the contract as everything will need to be deployed again at a different address if there's a code upgrade.

Remediation
It is recommended to create dynamic functions to address upgrades so that it becomes easier for developers to make changes at a later stage if necessary.
The said function should have proper access controls to make sure only administrators can call that function using access control modifiers.
There should also be a zero address validation in the function to make sure the tokens are not lost.
If the address is supposed to be hardcoded, it is advisable to make it a constant if its value is not getting updated.

Retest
Client Comment: Acknowledged

Bug ID #G001 [Acknowledged]

Public constants can be private

Vulnerability Type
Gas Optimization (SCWE-082)

Severity
Gas

Description
Public constant variables cost more gas because the EVM automatically creates getter functions for them and adds entries to the method ID table. The values can be read from the source code instead.

Affected Code

- 0x8c881e37b357B74F2C42a1Ae5d1c908078753923#code#F1#L10
- 0x8c881e37b357B74F2C42a1Ae5d1c908078753923#code#F1#L11
- 0x8c881e37b357B74F2C42a1Ae5d1c908078753923#code#F1#L12
- 0x8c881e37b357B74F2C42a1Ae5d1c908078753923#code#F1#L13
- 0x8c881e37b357B74F2C42a1Ae5d1c908078753923#code#F1#L14

Impacts
Public constants are more costly due to the default getter functions created for them, increasing the overall gas cost.

Remediation
If reading the values for the constants is not necessary, consider changing the public visibility to private.

Retest
Client Comment: Acknowledged

6. The Disclosure ----------

The Reports provided by CredShields are not an endorsement or condemnation of any specific project or team and do not guarantee the security of any specific project. The contents of this report are not intended to be used to make decisions about buying or selling tokens, products, services, or any other assets and should not be interpreted as such.

Emerging technologies such as Smart Contracts and Solidity carry a high level of technical risk and uncertainty. CredShields does not provide any warranty or representation about the quality of code, the business model or the proprietors of any such business model, or the legal compliance of any business. The report is not intended to be used as investment advice and should not be relied upon as such.

CredShields Audit team is not responsible for any decisions or actions taken by any third party based on the report.

Your Secure Future Starts Here

CRED SHIELDS

At CredShields, we're more than just auditors. We're your strategic partner in ensuring a secure Web3 future. Our commitment to your success extends beyond the report, offering ongoing support and guidance to protect your digital assets

AICPA SOC
aicpa.org/soc4so

SCS Advocates
OWASP
OWASP Smart Contract Top 10
Security Risks & Vulnerabilities

20A TANJONG PAGAR ROAD, SINGAPORE (088443)

info@credshields.com