Audited by

CredShields

Smart Contract Audit

November 3, 2025 • CONFIDENTIAL

Description

This document details the process and result of the Smart Contract audit performed by CredShields Technologies PTE. LTD. on behalf of Capx AI between October 28th, 2025, and October 29th, 2025. A retest was performed on October 30th, 2025.

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

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

Prepared for
Capx AI

Table of Contents

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

Capx AI engaged CredShields to perform a smart contract audit from October 28th, 2025, to October 29th, 2025. During this timeframe, 3 vulnerabilities were identified. **A retest was performed on October 30th, 2025, 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 "Capx AI" and should be prioritized for remediation; 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
Capx Onramp Bridge	0	0	0	2	0	1	**3**
	0	**0**	**0**	**2**	**0**	**1**	**3**

Table: Vulnerabilities Per Asset in Scope

The CredShields team conducted the security audit to focus on identifying vulnerabilities in Capx Onramp Bridge's scope during the testing window while abiding by the policies set forth by Capx AI'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 Capx AI'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 Capx AI can implement both manual and automated procedures to eliminate entire classes of vulnerabilities in the future. By taking a proactive approach, Capx AI can future-proof its security posture and protect its assets.

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

Capx AI engaged CredShields to perform the Capx Onramp Bridge 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 October 28th, 2025, to October 29th, 2025, 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
https://github.com/Capx-AI/Capx-OnRamp-Bridge/tree/136e9ed0ab77021b7b617f42d9d015a9bc4409bb

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

Capx AI 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, 3 security vulnerabilities were identified in the asset.

VULNERABILITY TITLE	SEVERITY	SCWE | Vulnerability Type
Outdated Pragma	Low	Outdated Compiler Version (SCWE-061)
Missing events in important functions	Low	Missing Best Practices
Cheaper Inequalities in if()	Gas	Gas Optimization (SCWE-082)

Table: Findings in Smart Contracts

4. Remediation Status ------------------

Capx AI is actively partnering with CredShields from this engagement to validate the discovered vulnerabilities' remediations. **A retest was performed on October 30th, 2025, and all the issues have been addressed.**

Also, the table shows the remediation status of each finding.

VULNERABILITY TITLE	SEVERITY	REMEDIATION STATUS
Outdated Pragma	Low	Won't Fix [Oct 30, 2025]
Missing events in important functions	Low	Fixed [Oct 30, 2025]
Cheaper Inequalities in if()	Gas	Won't Fix [Oct 30, 2025]

Table: Summary of findings and status of remediation

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

Bug ID #L001 [Acknowledged]

Outdated Pragma

Vulnerability Type
Outdated Compiler Version (SCWE-061)

Severity
Low

Description
The smart contract is using an outdated version of the Solidity compiler specified by the pragma directive i.e. 0.8.27. Solidity is actively developed, and new versions frequently include important security patches, bug fixes, and performance improvements. Using an outdated version exposes the contract to known vulnerabilities that have been addressed in later releases. Additionally, newer versions of Solidity often introduce new language features and optimizations that improve the overall security and efficiency of smart contracts.

Affected Code
- https://github.com/Capx-AI/Capx-OnRamp-Bridge/blob/136e9ed0ab77021b7b617f42d9d015a9bc4409bb/contracts/BridgeDestination.sol#L2
- https://github.com/Capx-AI/Capx-OnRamp-Bridge/blob/136e9ed0ab77021b7b617f42d9d015a9bc4409bb/contracts/BridgeSource.sol#L2

Impacts
The use of an outdated Solidity compiler version can have significant negative impacts. Security vulnerabilities that have been identified and patched in newer versions remain exploitable in the deployed contract.
Furthermore, missing out on performance improvements and new language features can result in inefficient code execution and higher gas costs.

Remediation
It is suggested to use the 0.8.29 pragma version.
Reference: https://scs.owasp.org/SCWE/SCSVS-CODE/SCWE-061/

Retest
Acknowledged - Deferred to next maintenance cycle

Bug ID #L002 [Fixed]

Missing events in important functions

Vulnerability Type
Missing Best Practices

Severity
Low

Description
Events are inheritable members of contracts. When you call them, they cause the arguments to be stored in the transaction's log—a special data structure in the blockchain. These logs are associated with the address of the contract which can then be used by developers and auditors to keep track of the transactions.

The contract was found to be missing these events on certain critical functions which would make it difficult or impossible to track these transactions off-chain.

Affected Code
- https://github.com/Capx-AI/Capx-OnRamp-Bridge/blob/136e9ed0ab77021b7b617f42d9d015a9bc4409bb/contracts/BridgeSource.sol#L809
- https://github.com/Capx-AI/Capx-OnRamp-Bridge/blob/136e9ed0ab77021b7b617f42d9d015a9bc4409bb/contracts/BridgeDestination.sol#L414

Impacts
Events are used to track the transactions off-chain and missing these events on critical functions makes it difficult to audit these logs if they're needed at a later stage.

Remediation
Consider emitting events for important functions to keep track of them.

Retest
This is fixed.

Bug ID #G001 [Won't Fix]

Cheaper Inequalities in if()

Vulnerability Type
Gas Optimization (SCWE-082)

Severity
Gas

Description
The contract was found to be doing comparisons using inequalities inside the "if" statement. When inside the "if" statements, non-strict inequalities (>=, <=) are usually cheaper than the strict equalities (>, <).

Affected Code

- https://github.com/Capx-AI/Capx-OnRamp-Bridge/blob/136e9ed0ab77021b7b617f42d9d015a9bc4409bb/contracts/BridgeSource.sol#L321
- https://github.com/Capx-AI/Capx-OnRamp-Bridge/blob/136e9ed0ab77021b7b617f42d9d015a9bc4409bb/contracts/BridgeSource.sol#L401
- https://github.com/Capx-AI/Capx-OnRamp-Bridge/blob/136e9ed0ab77021b7b617f42d9d015a9bc4409bb/contracts/BridgeSource.sol#L460
- https://github.com/Capx-AI/Capx-OnRamp-Bridge/blob/136e9ed0ab77021b7b617f42d9d015a9bc4409bb/contracts/BridgeSource.sol#L782
- https://github.com/Capx-AI/Capx-OnRamp-Bridge/blob/136e9ed0ab77021b7b617f42d9d015a9bc4409bb/contracts/BridgeSource.sol#L816
- https://github.com/Capx-AI/Capx-OnRamp-Bridge/blob/136e9ed0ab77021b7b617f42d9d015a9bc4409bb/contracts/BridgeDestination.sol#L337
- https://github.com/Capx-AI/Capx-OnRamp-Bridge/blob/136e9ed0ab77021b7b617f42d9d015a9bc4409bb/contracts/BridgeDestination.sol#L360
- https://github.com/Capx-AI/Capx-OnRamp-Bridge/blob/136e9ed0ab77021b7b617f42d9d015a9bc4409bb/contracts/BridgeDestination.sol#L395
- https://github.com/Capx-AI/Capx-OnRamp-Bridge/blob/136e9ed0ab77021b7b617f42d9d015a9bc4409bb/contracts/BridgeDestination.sol#L467

Impacts
Using strict inequalities inside "if" statements costs more gas.

Remediation
It is recommended to go through the code logic, and, **if possible**, modify the strict inequalities with the non-strict ones to save gas as long as the logic of the code is not affected.

Retest
Thank you for this suggestion. We've carefully evaluated this micro-optimization and concluded it's not appropriate for our use case.

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

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

Audited by