Audited by CRED SHIELDS

CredShields

Smart Contract Audit

October 2, 2025 • CONFIDENTIAL

Description

This document details the process and result of the Smart Contract audit performed by CredShields Technologies PTE. LTD. on behalf of Klink Finance between October 1st, 2025, and October 2nd, 2025. A retest was performed on October 2nd, 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
Klink Finance

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 6
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 ——————————- **10**
3.1 Findings Overview 10
3.1.1 Vulnerability Summary 10
4. Remediation Status ————————- **11**
5. Bug Reports ——————————— **12**
Bug ID #M001 [Won't Fix] 12
Owner Can Arbitrarily Extend Prelaunch Lock Period 12
Bug ID #L001 [Fixed] 13
Floating and Outdated Pragma 13
Bug ID #L002 [Fixed] 14
Use Ownable2Step 14
Bug ID #L003 [Fixed] 15
Missing Zero Address Validations 15
6. The Disclosure ——————————- **16**

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

Klink Finance engaged CredShields to perform a smart contract audit from October 1st, 2025, to October 2nd, 2025. During this timeframe, 4 vulnerabilities were identified. **A retest was performed on October 2nd, 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 "Klink Finance" 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
Klink Token Contract	0	0	1	3	0	0	**4**
	0	**0**	**1**	**3**	**0**	**0**	**4**

Table: Vulnerabilities Per Asset in Scope

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

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

Klink Finance engaged CredShields to perform a Klink Token Contract 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 1st, 2025, to October 2nd, 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
Audit Commit: https://github.com/KlinkFinance/klink-contracts/blob/97520ac6e15c2737b49025408ae11b41c5891aef/contracts/KlinkToken.sol **Retested/Audited Commit:** https://github.com/KlinkFinance/klink-contracts/blob/57a2635096ece4be19544b93e06460f680a61053/contracts/KlinkToken.sol **Deployed Token Contract:** https://bscscan.com/address/0x76E9b54B49739837bE8aD10c3687Fc6b543de852#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

Klink Finance 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	SCWE | Vulnerability Type
Owner Can Arbitrarily Extend Prelaunch Lock Period	Medium	Access Control (SCWE-016)
Floating and Outdated Pragma	Low	Floating Pragma (SCWE-060)
Use Ownable2Step	Low	Missing Best Practices
Missing Zero Address Validations	Low	Missing Input Validation (SC04-Lack Of Input Validation)

Table: Findings in Smart Contracts

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

Klink Finance is actively partnering with CredShields from this engagement to validate the discovered vulnerabilities' remediations. **A retest was performed on October 2nd, 2025, and all the issues have been addressed.**

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

VULNERABILITY TITLE	SEVERITY	REMEDIATION STATUS
Owner Can Arbitrarily Extend Prelaunch Lock Period	Medium	Won't Fix [October 2, 2025]
Floating and Outdated Pragma	Low	Fixed [October 2, 2025]
Use Ownable2Step	Low	Fixed [October 2, 2025]
Missing Zero Address Validations	Low	Fixed [October 2, 2025]

Table: Summary of findings and status of remediation

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

Bug ID #M001 [Won't Fix]

Owner Can Arbitrarily Extend Prelaunch Lock Period

Vulnerability Type
Access Control (SCWE-016)

Severity
Medium

Description
The contract defines a transferAllowedTimestamp used to control when token transfers become enabled. During pre-launch, this value is intended to be set once by the owner before trading begins, allowing the project to lock transfers until a defined launch time. However, the implementation permits multiple updates to transferAllowedTimestamp while still in pre-launch. This means the owner can continuously delay the unlock time at will. The root cause is the absence of a one-time restriction on setting transferAllowedTimestamp before transfers are enabled.

Affected Code
- https://github.com/KlinkFinance/klink-contracts/blob/97520ac6e15c2737b49025408ae11b41c5891aef/contracts/KlinkToken.sol#L77-L80

Impacts
Investors and users relying on a transparent launch date face uncertainty, as the owner can arbitrarily extend the lock period multiple times.

Remediation
It is recommended to restrict the transferAllowedTimestamp update to only one pre-launch change, ensuring it cannot be arbitrarily changed multiple times.

Retest
Client's Comments: We only want transferAllowedTimestamp to be used for the initial token launch. After transfers are live (i.e., once the timestamp has passed), this value will never be updated again. If there's a need to delay launch before that moment, we can set the timestamp later, but once the token is open for transfers it remains fixed permanently.

Bug ID #L001 [Fixed]

Floating and Outdated Pragma

Vulnerability Type
Floating Pragma (SCWE-060)

Severity
Low

Description
Locking the pragma helps ensure that the contracts do not accidentally get deployed using an older version of the Solidity compiler affected by vulnerabilities.
The contract allowed floating or unlocked pragma to be used, i.e., ^0.8.0. This allows the contracts to be compiled with all the solidity compiler versions above the limit specified. The following contracts were found to be affected -

Affected Code

- https://github.com/KlinkFinance/klink-contracts/blob/97520ac6e15c2737b49025408ae11b41c5891aef/contracts/KlinkToken.sol#L2

Impacts
If the smart contract gets compiled and deployed with an older or too recent version of the solidity compiler, there's a chance that it may get compromised due to the bugs present in the older versions or unidentified exploits in the new versions.
Incompatibility issues may also arise if the contract code does not support features in other compiler versions, therefore, breaking the logic.
The likelihood of exploitation is low.

Remediation
It is suggested that keep the compiler versions consistent in all the smart contract files. Do not allow floating pragmas anywhere. It is suggested to use the 0.8.29 pragma version
Reference: https://scs.owasp.org/SCWE/SCSVS-CODE/SCWE-060/

Retest
The pragma has been updated and fixed to 0.8.30.

Bug ID #L002 [Fixed]

Use Ownable2Step

Vulnerability Type
Missing Best Practices

Severity
Low

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
- https://github.com/KlinkFinance/klink-contracts/blob/97520ac6e15c2737b49025408ae11b41c5891aef/contracts/KlinkToken.sol#L17

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
This is fixed by using ownable2step.

Bug ID #L003 [Fixed]

Missing Zero Address Validations

Vulnerability Type
Missing Input Validation (SC04-Lack Of Input Validation)

Severity
Low

Description:
The contracts were found to be setting new addresses without proper validations for zero addresses.
Address type parameters should include a zero-address check otherwise contract functionality may become inaccessible or tokens burned forever.
Depending on the logic of the contract, this could prove fatal and the users or the contracts could lose their funds, or the ownership of the contract could be lost forever.

Affected Code

- https://github.com/KlinkFinance/klink-contracts/blob/97520ac6e15c2737b49025408ae11b41c5891aef/contracts/KlinkToken.sol#L2
- https://github.com/KlinkFinance/klink-contracts/blob/97520ac6e15c2737b49025408ae11b41c5891aef/contracts/KlinkToken.sol#L93
- https://github.com/KlinkFinance/klink-contracts/blob/97520ac6e15c2737b49025408ae11b41c5891aef/contracts/KlinkToken.sol#L99
- https://github.com/KlinkFinance/klink-contracts/blob/97520ac6e15c2737b49025408ae11b41c5891aef/contracts/KlinkToken.sol#L104

Impacts
If address type parameters do not include a zero-address check, contract functionality may become unavailable or tokens may be burned permanently.

Remediation
To fix this add a zero address validation to all the functions where addresses are being set.

Retest
This has been fixed.

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

Audited by CRED SHIELDS

20A TANJONG PAGAR ROAD, SINGAPORE (088443)

info@credshields.com