MixBytes()

Keep3r Sunset Security Audit Report

MAY 27, 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

This contract facilitates the distribution of a fixed amount of ERC20 tokens (USDC as it is stated in the README) to holders of the vKP3R token, based on their balances at a specific snapshot block. The contract ensures that each eligible address (locker) can claim their share only once, proportionally calculated using their snapshot balance relative to the total supply at that block.

The audit was conducted over 1 day by 3 auditors.

During the audit, in addition to checking well-known attack vectors and those listed in our internal checklist, we carefully investigated the following:

Snapshot-based distribution mechanism. The contract uses a snapshot-based approach to determine user balances and total supply of vKP3R at a specific block number. The use of `balanceOfAt`, `totalSupplyAt` functions at the snapshot block and `has_claimed` mapping is particularly important as it prevents double distribution of rewards for the same tokens, ensuring that the total amount of rewards distributed cannot exceed the intended amount.

Clawback functionality and security. The contract implements a clawback mechanism that allows recovery of unclaimed tokens after a two-year period. This feature was thoroughly analyzed for its security implications, particularly focusing on the ability to clawback any ERC20 token from the contract. The implementation uses a fixed `FUNDER` address, which while providing certainty, also introduces some inflexibility in the system. The two-year waiting period for the main distribution token acts as a safety mechanism, ensuring that users have ample time to claim their rewards before any clawback can occur. It's worth noting that the clawback function can be used to transfer any ERC20 token from the contract to the `FUNDER` address. While this is a legitimate feature for recovering unclaimed rewards, it could potentially be used in social engineering attacks if malicious tokens are sent to the contract. However, this risk is mitigated by the use of a multisig for the `FUNDER` address, making such attacks highly improbable.

Precision and dust handling. The distribution calculation involves division operations that could lead to precision loss. Our analysis showed that due to integer division in the calculation of claimable amounts (formula `TOTAL_AMOUNT * balanceOfAt // totalSupplyAt`), individual users might experience a precision loss of up to 1 USDC. While this precision loss exists, it doesn't prevent users from claiming their rewards. The dust amounts are effectively capped by the number of claimers, with approximately 670 claimers resulting in a maximum possible accumulation of dust that would be claimable through the clawback mechanism, ensuring no funds are permanently locked in the contract.

Claim process security. The claim functionality was thoroughly reviewed for its security properties. The implementation includes protection against double claims through a has_claimed mapping and proper balance tracking through the snapshot mechanism. However, there is a potential issue with the interaction between clawback and claim functions. After the clawback has been executed, users can still call the claim function, but it will revert without any informative error message due to an attempt to transfer non-existent tokens. This could be improved by adding a check for the contract's token balance before attempting the transfer.

The main focus of the audit was to review the snapshot distribution mechanism for rKP3R rewards (represented by USDC) to vKP3R lockers, ensuring accurate proportional allocation based on snapshot balances and secure one-time claims per eligible address. Key areas of examination included the correctness of balance calculations using historical data (balanceOfAt and totalSupplyAt), the integrity of the claim logic, and proper handling of unclaimed tokens through a clawback mechanism. Notably, the vKP3R token contract itself was out of the audit scope, and its correctness was assumed for the purposes of the distribution logic review.

The overall code quality was clear, well-structured, and aligned with Vyper best practices, with minimal complexity and a focus on safety and transparency.

1.3 Project Overview

Summary

Title	Description
Client Name	Keep3r
Project Name	Keep3r Sunset
Type	Vyper
Platform	EVM
Timeline	09.05.2025 - 15.05.2025

Scope of Audit

File	Link
contracts/snapshot_distribution.vy	-

Versions Log

Date	Commit Hash	Note
09.05.2025	08cf93aa20f3e18e6290af0e2dac7440287bba686e6ae032292e5897971261b6	Initial Hash of the Scope
15.05.2025	45a311170f7e56aeca99e426c084bf06e9204ce741815f9fd186b2158ec552d6	Hash of the Scope for Re-audit

Mainnet Deployments

File	Address	Blockchain
snapshot_distribution.vy	0xea4021...38d30967	Ethereum Mainnet

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	0
Low	3

Findings Statuses

ID	Finding	Severity	Status
L-1	Missing Validations in Constructor	Low	Fixed
L-2	Incorrect Clawback Timeout Duration	Low	Fixed
L-3	Incorrect Naming: vKR3R in Comments	Low	Fixed

2. Findings Report

2.1 Critical

Not Found

2.2 High

Not Found

2.3 Medium

Not Found

2.4 Low

L-1	Missing Validations in Constructor		
Severity	Low	**Status**	Fixed in 45a311170f7e56aeca99e426c084bf06e9204ce741815f9fd186b2158ec552d6

Description
This issue has been identified within the __init__ function of the vKP3R Snapshot Distribution contract.
The constructor lacks validations for input parameters, even though the contract is tailored for a specific airdrop. In particular, the TOKEN address is not checked for being the zero address, and the SNAPSHOT block number is not validated to ensure it is not set in the future.
The issue is classified as **Low** severity because these missing checks, although not immediately exploitable in the current context, could lead to future misconfigurations or vulnerabilities if the contract is reused or modified.

Recommendation
We recommend adding explicit validations in the constructor to ensure that the TOKEN address is not the zero address and that the SNAPSHOT block number is not set in the future.

L-2	Incorrect Clawback Timeout Duration		
Severity	Low	**Status**	Fixed in 45a311170f7e56aeca99e426c084bf06e9204ce741815f9fd186b2158ec552d6

Description

The clawback timeout is set to `2 * 356 days` instead of the standard `2 * 365 days` (2 years), which may cause confusion or misalignment with expected behavior based on common yearly time assumptions. This could result in premature clawback availability.

Recommendation

We recommend updating the clawback timeout duration calculation to use 365 days (i.e., `2 * 365 * 24 * 60 * 60` seconds) to align with standard calendar expectations.

L-3	Incorrect Naming: vKR3R in Comments		
Severity	Low	**Status**	Fixed in 45a311170f7e56aeca99e426c084bf06e9204ce741815f9fd186b2158ec552d6

Description
There are typos in comments referencing vKR3R, which is incorrect and may cause confusion for reviewers or developers trying to understand the code. The correct token name is vKP3R.

Recommendation
We recommend fixing the comments to reflect the correct name (vKP3R) to improve clarity.

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