Drips contracts

Security Review

Cantina Managed review by:

0xIcingdeath, Lead Security Researcher
High Byte, Security Researcher

April 29, 2025

Contents

1 Introduction

1.1 About Cantina

Cantina is a security services marketplace that connects top security researchers and solutions with clients. Learn more at cantina.xyz

1.2 Disclaimer

Cantina Managed provides a detailed evaluation of the security posture of the code at a particular moment based on the information available at the time of the review. While Cantina Managed endeavors to identify and disclose all potential security issues, it cannot guarantee that every vulnerability will be detected or that the code will be entirely secure against all possible attacks. The assessment is conducted based on the specific commit and version of the code provided. Any subsequent modifications to the code may introduce new vulnerabilities that were absent during the initial review. Therefore, any changes made to the code require a new security review to ensure that the code remains secure. Please be advised that the Cantina Managed security review is not a replacement for continuous security measures such as penetration testing, vulnerability scanning, and regular code reviews.

1.3 Risk assessment

Severity	Description
Critical	*Must* fix as soon as possible (if already deployed).
High	Leads to a loss of a significant portion (>10%) of assets in the protocol, or significant harm to a majority of users.
Medium	Global losses <10% or losses to only a subset of users, but still unacceptable.
Low	Losses will be annoying but bearable. Applies to things like griefing attacks that can be easily repaired or even gas inefficiencies.
Gas Optimization	Suggestions around gas saving practices.
Informational	Suggestions around best practices or readability.

1.3.1 Severity Classification

The severity of security issues found during the security review is categorized based on the above table. Critical findings have a high likelihood of being exploited and must be addressed immediately. High findings are almost certain to occur, easy to perform, or not easy but highly incentivized thus must be fixed as soon as possible.

Medium findings are conditionally possible or incentivized but are still relatively likely to occur and should be addressed. Low findings a rare combination of circumstances to exploit, or offer little to no incentive to exploit but are recommended to be addressed.

Lastly, some findings might represent objective improvements that should be addressed but do not impact the project's overall security (Gas and Informational findings).

2 Security Review Summary

Drips is a protocol and app built on Ethereum that enables organizations and individuals to directly and publicly provide funding to the free and open source software projects they depend on the most.

Drips also includes gas-optimized and integrated primitives for streaming and splitting tokens, allowing users and web3 apps to stream and split funds by the second with continuous settlement for use cases like contributor payments, vesting and subscription memberships.

From Mar 28th to Mar 29th the Cantina team conducted a review of drips-contracts on commit hash 42587531. The team identified a total of **3** issues:

Issues Found

Severity	Count	Fixed	Acknowledged
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	0	0	0
Low Risk	0	0	0
Gas Optimizations	0	0	0
Informational	3	1	2
Total	**3**	**1**	**2**

3 Findings

3.1 Informational

3.1.1 Missing zero address check to ensure unwrapper can't accidentally burn tokens

Severity: Informational

Context: NativeTokenUnwrapper.sol#L36-L43

Description: The `unwrap` function is missing a check to ensure that the `recipient` address is not `address(0)`. If it is, the native token can be accidentally burnt when withdrawn.

Recommendation: Add a `require (recipient != address(0))` to the `unwrap` function.

Drips: Acknowledged, won't fix. There are many places in the smart contracts API where passing an invalid address will lead to loss of funds. I agree that the zero address is slightly easier to accidentally pass than any other invalid address, but IMO this little protection isn't worth more logic in the contract and a more complicated API surface.

Cantina Managed: Acknowledged.

3.1.2 ZKSync Integration Recommendations

Severity: Informational

Context: *(No context files were provided by the reviewer)*

Description: As part of the review, the client asked us specifically to look at one commit that replaced L1's use of `CloneProxies` with a precompile call in ZKSync to `create2` directly. The below compiles our recommendations on future maintainability:.

- Add a chain ID check on the contracts to differentiate whether the deploy is on Ethereum or ZKSync. Implement separate cases for both, and revert if there is an attempt to be used with an unsupported chain.
- Document the limitations of ZKSync that require these workarounds.
- If additional chains are being deployed, consider *either* creating external libraries that contain deployment logic and deciding which logic to execute *or* ensuring that the list of supported chain ID's is kept up to date and monitored.

Note: Issue is TBD and we are still investigating what additional recommendations we may be able to provide.

Drips: Those are fair points. We don't plan to merge the zkSync branch to the regular branch, they will be kept in separate branches for the foreseeable future. If there are any new features or fixes, they can be backported and tested for zkSync compatibility.

The reason is that there are too many differences in not only the production code but also in the tests and the deployment, so it could needlessly complicate the project. The zkSync compiler also has limitations and sometimes can't even process valid code accepted by solc. The only way to find it out is to keep cross-compiling and applying fixes and refactorings which is a rather large overhead when developing.

Cantina Managed: Acknowledged.

3.1.3 Add documentation on the payable receive function

Severity: Informational

Context: NativeTokenUnwrapper.sol#L29

Description: The codebase maintains a fairly high standard of documenting security-related assumptions in the Natspec. We recommend to document the impacts of the payable receive function on the unwrapper contract as well, namely - If a user accidentally transfers ETH to the contract without an associated native token, the funds will be lost, as a user can only take out at most the `amount` of native tokens that have been transferred.

Recommendation: Add a natspec comment or in user documentation warning users from attempting to deposit ether directly into the `NativeTokenUnwrapper` contract.

Drips: Good idea. Alternatively `unwrap` could transfer not the amount of wrapped tokens that were unwrapped, but all native tokens held after the unwrapping. Either way, documentation on `receive` would be useful. Fixed in commit 80ec1104.

Cantina Managed: Fix verified.