ENIGMA DARK

Securing the Shadows

Security Review Extension

Flaunch

December 7, 2024

Contents

Summary

Enigma Dark
Enigma Dark is a web3 security firm leveraging the best talent in the space to secure all kinds of blockchain protocols and decentralized apps. Our team comprises experts who have honed their skills at some of the best auditing companies in the industry. With a proven track record as highly skilled white-hats, they bring a wealth of experience and a deep understanding of the technology and the ecosystem.

Learn more about us at enigmadark.com

Flaunch
The ƒlaunch protocol is a launchpad platform built on Uniswap v4, incorporating advanced mechanics for token launch and trading. It emphasizes sustainability in token economies by introducing features such as Progressive Bid Walls and decentralized revenue-sharing models. These mechanisms aim to create a fair and transparent environment for participants, balancing incentives for developers and traders to prioritize long-term ecosystem stability over short-term speculation.

Engagement Overview

Over the course of 3 days starting December 2nd 2024, the Enigma Dark team conducted a security review extension of the Flaunch project, reviewing the diff between the commits listed below. The review was performed by two Lead Security Researchers, vnmrtz & 0xWeiss.

The following repositories were reviewed at the specified commits:

Repository	Commit
flayerlabs/flaunch-contracts	`diff 950924e..488ad9c`

Risk Classification

Severity	Description
Critical	Vulnerabilities that lead to a loss of a significant portion of funds of the system.
High	Exploitable, causing loss or manipulation of assets or data.
Medium	Risk of future exploits that may or may not impact the smart contract execution.
Low	Minor code errors that may or may not impact the smart contract execution.
Informational	Non-critical observations or suggestions for improving code quality, readability, or best practices.

Vulnerability Summary

Severity	Count	Fixed	Acknowledged
Critical	1	1	0
High	0	0	0
Medium	0	0	0
Low	0	0	0
Informational	1	1	0

Findings

Index	Issue Title	Status
C-01	Creator fees can underflow	Fixed
I-01	Miscellaneous	Fixed

Detailed Findings

Critical Risk

C-01 - Creator fees can underflow

Severity: Critical Risk

Technical Details:

The `creatorFeeAllocation` which can be set by the creator, has not upper limit. The `creatorFeeAllocation` variable gets stored by poolId:

```
  if (_params.creatorFeeAllocation != 0) {
            creatorFee[poolId] = _params.creatorFeeAllocation;
        }
```

after, when calculating fees `_creatorFee` is used. If `_creatorFee` is bigger than `ONE_HUNDRED_PERCENT`, then `creator_` would be bigger than `_amount` and would underflow. Such fee can be pre-calculated to underflow to the exact wanted number:

```
// The creator is now given their share
        uint24 _creatorFee = creatorFee[_poolId];
        if (_creatorFee != 0) {
            unchecked {
                creator_ = _amount * _creatorFee / ONE_HUNDRED_PERCENT;

                // Reduce the amount available for the following lions
                _amount -= creator_;
            }
        }
```

Impact:

`_amount` can underflow which will miss-calculate all fees and potentially drain the contract

Recommendation:

Add a max cap to the creator fees. Such cap ideally should be a variable that can be updated through a permissioned function. Additionally, remove any unchecked that it is not strictly necessary.

Developer Response:
We have added back in the creator fee allocation check at 100%, fixed at commit `5b12258`.

High Risk

No issues found.

Medium Risk

No issues found.

Low Risk

No issues found.

Informational

I-01 - Miscellaneous

Severity: Informational

Technical Details:

- Most of the variables such as the launch period are immutable, but as seen from v1 to v1.1, they have drastically changed. Better to make the key variables that can be potentially updated mutable and add a permissioned setter function to update them in the future.
- Remove unchecked blocks that are not really needed. Given a previous finding, we advise to be careful when using those.
- Reuse `poold` variable when deleting the `flaunchesAt` value in `beforeSwap`, instead of calculating the pool id again: `_key.toId()`
- The `flaunchAt` parameter is poorly named, as it now represents a delay relative to the current timestamp rather than the specific timestamp at which the flaunch is scheduled to occur. Consider renaming the parameter and its references to something more descriptive, such as `flaunchDelay`.

Recommendation:

Fix the above mentioned issues

Developer Response:

- Removed unchecked to keep under/over-flow checks, fixed at commit `864757b`.
- Reuse poolId, fixed at commit `c7deb6a`.
- Updated the `flaunchesAt` application to use a timestamp, rather than duration. Only scheduling the flaunch if it's in the future, fixed at commit `a298120`.

Disclaimer

This report does not endorse or critique any specific project or team. It does not assess the economic value or viability of any product or asset developed by parties engaging Enigma Dark for security assessments. We do not provide warranties regarding the bug-free nature of analyzed technology or make judgments on its business model, proprietors, or legal compliance.

This report is not intended for investment decisions or project participation guidance. Enigma Dark aims to improve code quality and mitigate risks associated with blockchain technology and cryptographic tokens through rigorous assessments.

Blockchain technology and cryptographic assets inherently involve significant risks. Each entity is responsible for conducting their own due diligence and maintaining security measures. Our assessments aim to reduce vulnerabilities but do not guarantee the security or functionality of the technologies analyzed.

This security engagement does not guarantee against a hack. It is a review of the codebase at a during a specific period of time. Enigma Dark makes no warranties regarding the security of the code and does not warrant that the code is free from defects. By deploying or using the code, the project and users of the contracts agree to use the code at their own risk. Any modifications to the code will require a new security review.