ENIGMA DARK

Securing the Shadows



Security Review

Arrakis Modular
PancakeSwap V3 Integration

Sept, 2025

Contents

1. Summary
2. Engagement Overview
3. Risk Classification
4. Vulnerability Summary
5. Findings
6. Disclaimer

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

Arrakis Modular: PancakeSwap V3 Integration
Arrakis Modular is a DEX-agnostic liquidity management framework built around Meta-Vaults and standardized modules. It enables integration with any two-token DEX (like Uniswap V4, Balancer, Ambient) through reusable, and upgradeable components. Designed for flexibility and scalability, it supports both public and private vaults, simplifies liquidity provisioning, and lays the groundwork for advanced strategies across DeFi.

Engagement Overview

Over the course of 1.6 weeks (1 week and 3 days), beginning September 15 2025, the Enigma Dark team conducted a security review of the Arrakis Modular: PancakeSwap V3 Integration project. The review was performed by one Lead Security Researcher and one Security Researcher.

The following repositories were reviewed at the specified commits:

Repository	Commit
ArrakisFinance/arrakis-modular	`431882bcc4dba938e0cd74f19d8a86146a20d480`

The scope of the review covered the following files:

```
src
├── abstracts
|   └── PancakeSwapV3StandardModule
├── libraries
|   └── UnderlyingV3
├── modules
|   └── PancakeSwapV3StandardModulePrivate
└── structs
    ├── SUniswapV3
    └── SPancakeSwapV3
```

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	0	0	0
High	0	0	0
Medium	0	0	0
Low	5	3	2
Informational	5	3	2

Findings

Index	Issue Title	Status
L-01	Deviation Can Slightly Exceed Threshold	Fixed
L-02	Manager could slowly drain funds by swapping on a custom router	Acknowledged
L-03	Missing storage gap in PancakeSwapV3StandardModule	Fixed
L-04	Arbitrary Users Can Move Unbounded Amounts From Reserves to LP	Acknowledged
L-05	Emergency Mode In MasterChef Skips Reward Growth Collection	Fixed
I-01	Missing Oracle Price Zero Check In Rebalance Validation	Fixed
I-02	Unclaimed cake rewards will be sent to the wrong receiver if their address is updated	Acknowledged
I-03	Redundant Oracle Price Calls In Slippage Check	Fixed
I-04	TotalUnderlyingAtPrice Ignores Fees Earned From Price Movement	Acknowledged
I-05	Potentially Unnecessary Slot0 Read Outside Increase Position Check	Fixed

Detailed Findings

High Risk

No issues found.

Medium Risk

No issues found.

Low Risk

L-01 - Deviation Can Slightly Exceed Threshold

Severity: Low Risk

Context:

- PancakeSwapV3StandardModule.sol#L978

Technical Details:
The `validateRebalance` function calculates price deviation using `FullMath.mulDiv` which rounds down. When this calculated deviation is compared against `maxDeviation`, rounding down means the actual deviation could exceed the maximum but down to the the prior mentioned rounding the `deviation` will only equal the `maxDeviation`. In such a case the check will pass and the rebalance will be validated as a valid rebalance despite the `maxDeviation` being exceeded.

Impact:
Rebalances can execute with price deviations higher than the maximum allowed threshold.

Recommendation:
Use `FullMath.mulDivRoundingUp` instead of `mulDiv` when calculating the deviation to ensure proper enforcement of maximum deviation limits.

Developer Response:
Fixed at commit `2a87960`.

L-02 - Manager could slowly drain funds by swapping on a custom router

Severity: Low Risk

Context:

- PancakeSwapV3StandardModule.sol#L452

Technical Details:
As specified by the Arrakis team, the manager role in the `Set Module` and `Rebalance` functions should be semi-trusted.

This means that any function access controlled to this role needs to be properly validated and bulletproof so that there is no way to maliciously impact the protocol by such role.

In the `rebalance` function, the manager would be able to steal the money in the contract using a fake router, as there is no whitelisted routers, just some basic validation:

```
if (
                params_.swapPayload.router == address(metaVault)
                    || params_.swapPayload.router == nftPositionManager
                    || params_.swapPayload.router == masterChefV3
                    || params_.swapPayload.router == CAKE
                    || params_.swapPayload.router == address(token0)
                    || params_.swapPayload.router == address(token1)
            ) {
                revert WrongRouter();
            }
```

The manager would be able to pass a custom router and always pass the lowest `expectedMinReturn` possible according to the maximum slippage specified and slowly drain the contract.

Impact:
The manager could drain the contract by swapping on a fake router.

Recommendation:
The `onlyMetaVaultOwner` should be able to whitelist certain routers for the manager to use. This router address should be then validated inside the rebalance function to be inside the whitelist. That way "chunky" validation above could also be removed as non of those addresses but be whitelisted as routers.

Developer Response:
Acknowledged. We have a cooldown period settable on the ArrakisStandardManager level, so rebalance cannot happen too frequently.

L-03 - Missing storage gap in PancakeSwapV3StandardModule

Severity: Low Risk

Context:

- PancakeSwapV3StandardModule.sol#L77

Technical Details:
The PancakeSwapV3StandardModule contract defines state variables such as:

```
    uint24 public maxSlippage;

    address public cakeReceiver;

    address public pool;

    IOracleWrapper public oracle;

    IArrakisMetaVault public metaVault;

    uint256 public managerFeePIPS;

    IERC20Metadata public token0;

    IERC20Metadata public token1;
```

However, it does not implement a storage gap or namespace layout as recommended by best practices for upgradeable contracts.

Impact:
Potential storage collisions in future upgrades

Recommendation:
Add a `__gap` array with a length of 50 - the slots occupied by currently declared storage variables:

```
    uint256[50] internal __gap;
```

Developer Response:
Fixed at commit `adb2088`.

L-04 - Arbitrary Users Can Move Unbounded Amounts From Reserves to LP

Severity: Low Risk

Context:

- PancakeSwapV3StandardModule.sol#L1255
- PancakeSwapV3StandardModule.sol#L1258

Technical Details:
The `_increaseLiquidity` function calculates both token amounts (`amt0` and `amt1`) based on position liquidity without maximum limits. Anyone can directly increase the position's liquidity through MasterChefV3's `increaseLiquidity` function due to it being a permissionless function. Which will inflate the values returned by `_principal` . When these inflated values are used in proportion calculations, it results in larger than expected amounts being pulled from reserves.

This griefing attack can become particularly impactful when positions are outside the current price range, as the liquidity position provides no fee revenue and cannot be used in other positions if reserves are to be utilized in the future. This also limits any integrators abilities to ensure its users that they will keep a certain amount of reserves at all times. While the excess liquidity can be decreased later, arbitrary users currently can dictate how much liquidity is moved from Arrakis to liquidity positions.

Impact:
Anyone can influence the amounts of token moved from reserves into unproductive positions, preventing the funds from being used elsewhere and disrupting protocol operations.

Recommendation:
Add maximum amount checks for both `amt0` and `amt1` to ensure only acceptable amounts can be moved during liquidity increases.

Developer Response:
Acknowledged. We appreciate the detailed pre-conditions for this attack to happen. We will add to our documentation, so our strategies will take it into account.

L-05 - Emergency Mode In MasterChef Skips Reward Growth Collection

Severity: Low Risk

Context:

- PancakeSwapV3StandardModule.sol#L743

Technical Details:
When MasterChefV3's emergency mode is active, it skips reward growth calculations during the withdraw. This becomes particularly problematic in the `withdrawManagerBalance` and `rebalance` functions. The reason being, is that whenever a position is unstaked the CAKE from the reward growth delta is not accounted for, where even with restaking the reward growth will be stamped to the current value. This means all rewards that were being tracked in MasterChef via reward growth are lost permanently. The issue stems from MasterChefV3's emergency mode behavior being silently accepted rather than being treated as an exceptional state that would prevent complete reward collection.

Impact:
Some rewards are permanently lost while MasterChefV3 is in emergency mode.

Recommendation:
Consider documenting this edge case so that users know that an accumulation of pending CAKE rewards could be lost if MasterChefV3 went into emergency mode.

Developer Response:
Fixed, this edge case will be added to the documentation.

Informational

I-01 - Missing Oracle Price Zero Check In Rebalance Validation

Severity: Informational

Context:

- PancakeSwapV3StandardModule.sol#L953

Technical Details:
The `validateRebalance` function retrieves oracle prices without validating that they are non-zero. While `_checkMinReturn` includes this validation, it is only called during swap operations. When a rebalance contains no swaps, the oracle price validation is skipped entirely. This allows rebalance validations to proceed with potentially invalid oracle prices, as long as no swaps are included in the operation.

Recommendation:
Add zero price validation check in `validateRebalance` to ensure oracle prices are valid regardless of rebalance operation type.

Developer Response:
Fixed at commit `2a87960`.

I-02 - Unclaimed cake rewards will be sent to the wrong receiver if their address is updated

Severity: Informational

Context:

- PancakeSwapV3StandardModule.sol#L875

Technical Details:
The `setReceiver` function allows the manager or the owner to change the receiver address for the cake rewards, also known as the manager balance. If the receiver address is changed to any other address, The unharvested rewards will not be claimed to the current receiver event though they technically belong to that address as they have been accrued while the address was the receiver. Instead, all the unharvested cake rewards will be sent to the new receiver address.

Impact:
Old receiver will lose the unharvested cake rewards as they will be sent to the new reciever.

Recommendation:
Call the `claimManager` function before updating the `cakeReceiver` address. Additionally, it would be good to add a boolean flag in case the address is compromised and the team wants to withdraw old rewards to a new address:

```
function setReceiver(
        address newReceiver_
    ) external whenNotPaused {
        address manager = metaVault.manager();

        if (IOwnable(manager).owner() != msg.sender) {
            revert OnlyManagerOwner();
        }

        address oldReceiver = cakeReceiver;
        if (newReceiver_ == address(0)) {
            revert AddressZero();
        }

        if (oldReceiver == newReceiver_) {
            revert SameReceiver();
        }
+     if(claimOldManager){ //boolean to be updated by the team in case old
receiver address is compromised
+      claimManager();
+     };
        cakeReceiver = newReceiver_;

        emit LogSetReceiver(oldReceiver, newReceiver_);
    }
```

Developer Response:
Acknowledged. Receiver will be updated by Arrakis Finance. We do prefer that the reward will be distributed to the new receiver, in case the old one is compromised.

I-03 - Redundant Oracle Price Calls In Slippage Check

Severity: Informational

Context:

- PancakeSwapV3StandardModule.sol#L1441
- PancakeSwapV3StandardModule.sol#L1454

Technical Details:
The `_checkMinReturn` function makes redundant oracle price calls when performing slippage checks. For both token0 and token1 paths, the function first stores the oracle price in the `oraclePrice` variable, but then makes another identical price call when calculating the minimum return. This results in unnecessary external calls that increase gas costs without providing additional validation.

Recommendation:
Use the already stored `oraclePrice` variable instead of making additional oracle price calls in the slippage check calculations.

Developer Response:
Fixed at commit `ae4d524`.

I-04 - TotalUnderlyingAtPrice Ignores Fees Earned From Price Movement

Severity: Informational

Context:

- PancakeSwapV3StandardModule.sol#L972

Technical Details:
The `totalUnderlyingAtPrice` function calculates position values at an arbitrary price point but does not account for fees that would be earned as the price moves from the current price to the target price. The function only considers the position's current fees and liquidity composition at the target price.

This creates a discrepancy between the reported underlying value and the actual value that would be realized when the price reaches that level, as any fees earned during the price movement are excluded from the calculation.

Recommendation:
Add documentation clearly stating that the function excludes fees that would be earned during price movement to the target price point.

Developer Response:
Acknowledged.

I-05 - Potentially Unnecessary Slot0 Read Outside Increase Position Check

Severity: Informational

Context:

- PancakeSwapV3StandardModule.sol#L621

Technical Details:
The `rebalance` function reads the pool's `slot0` data before checking if there are any increase positions to process. The `sqrtPriceX96` value obtained from this read is only used within the increase positions block that is guarded by a `length > 0` check. When a rebalance operation contains no increase positions the return value from the `slot0` function call is not used.

Recommendation:
Move the `slot0` read inside the increase positions length check to avoid unnecessary storage reads when no increase positions exist.

Developer Response:
Fixed at commit `d7d8c60`.

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 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.