

STATE MIND

Arrakis v2 core and palm

01-11-2022 - 25-11-2022

Table of contents



1. Project Brief		3
2. Finding Severity breakdown		5
3. Summary of findings		6
4. Conclusion		6
5. Findings report		8
High	Possible loss of funds by the manager	8
	Possible uniswap pool manipulation	8
	Possible fee hijacking (or DOS) by the manager	9
	Liquidity withdrawal from Uni position to Vault	9
	First minter can change LP token pricing	9
	Vault can renew the term for free	9
	Vault owner can lose tokens when increasing liquidity	10
	Rebalance DoS	10
Medium	Checking caller	10
	No check if liquidity > 0 in standardBurnParams()	10
	Anyone can call renewTerm()	11

Informational	Redundant modifier	11
	Unused modifier code duplication	11
	Reducing SLOAD operations	11
	Event indexed fields	12
	Typo in function naming	12
	Direct token transfers	12
	Range existence check	12
	Gas optimisation in range deletion	12
	Zero address check	13
	Bad readability	13
	Redundant check	13
	Gas optimisation in vaults()	14
	Gas optimisation in conversion from int to string	14
	Code refactoring	14
	Zero liquidity check	14
	Blacklisting strategy	14
	Redundant approve call	15
	Unmatched to documentation	15
	No max limit for ranges	15
	Variable totalSupply shadows function	15
	Possible to use LP token as vault token	16
	Possible to burn zero LP tokens	16
	Incorrect event emit	16
	TODO comments	16
	Using a number literal	17
	Inadequate view functions in ArrakisV2Storage	17

Function removePools() doesn't remove ranges	17
Functions can be declared as external	18
Function vaults() may run out of gas	18
Unnecessary external calls	18
Inefficient function ranges() in ArrakisV2Helper	19
Array operators can be changed to EnumerableSet.AddressSet	19
gelatoFeeCollector can be set to address(O)	19
Can fund vault balance with msg.value = O	19
Irrelevant comments	19
User can mint LP tokens to themselves in PALMTerms	20
setRestrictedMint() can be frontrun	20
Gas optimization in renewTerm()	20
User can receive less than expected when decreasing liquidity	20
Incorrect version	21
Gas optimizations in for loop	21
Code duplicating	21

7. Appendix B. Slither	22
------------------------	----

8. Appendix C. Tests	28
----------------------	----

1. Project Brief



Title	Description
Client	Arrakis
Project name	Arrakis v2 core and palm
Timeline	01-11-2022 - 25-11-2022
Initial commit	376bfcec803f0644fdc601db3a5772d2179c13aO, 06f8439430e3bOaf9cbbf887926ff93844c28a7d
Final commit	f4d6ae6eec553e9b316a64738306cee8f5909a81, 7102d4ca7f275000a47ce05ba7c7fd32fc1dfOOe

Short Overview

Arrakis is web3's liquidity layer, which at its core acts as a decentralized market-making platform enabling projects to create deep liquidity for their tokens on decentralized exchanges.

The core contracts allow users to:

- create an **ArrakisV2** vault instance that manages holdings of a given token pair
- dispatch and collect these holdings to/from Uniswap V3 Liquidity Positions (for the defined token pair) via a settable **manager** smart contract
- configure important vault setup parameters (manager, restrictedMint, pools) via the vault **owner** role























PALM is the first application built on top of the flexible ArrakisV2 core system, optimized for automated management of protocol owned liquidity (thus, Protocol Automated Liquidity Management).

PALM allows users to:

- Create a "private" vault that is managed by **PALMManager** who will run automated strategies on behalf of the vault creator. Only vault creators can add and remove liquidity from their private vault
- Vault creators have the ability to pick from a list of whitelisted strategy templates, and further configure the strategy with custom parameters
- Vault creators can increase or decrease liquidity deposited in the vault at any time, as well as change the strategy configuration (or delegate this strategy configuration ability to a third party)
- Finally vault creators can remove all of their liquidity and close the vault at any time

Project Scope

The audit covered the following files:

 ArrakisV2Storage.sol	 ArrakisV2FactoryStorage.sol	 ArrakisV2Helper.sol
 Position.sol	 UniswapV3Amounts.sol	 Underlying.sol
 Pool.sol	 Manager.sol	 SArrakisV2Helper.sol
 SArrakisV2.sol	 ArrakisV2Resolver.sol	 ArrakisV2.sol
 FArrakisV2Factory.sol	 ArrakisV2Factory.sol	 ArrakisV2Beacon.sol
 PALMManager.sol	 PALMTermsStorage.sol	 PALMManagerStorage.sol
 SPALMTerms.sol	 SPALMManager.sol	 FPALMTerms.sol
 PALMTerms.sol		

2. Finding Severity breakdown



All vulnerabilities discovered during the audit are classified based on its potential severity and has the following classification:

Severity	Description
Critical	Bugs leading to assets theft, fund access locking, or any other loss funds to be transferred to any party.
High	Bugs that can trigger a contract failure. Further recovery is possible only by manual modification of the contract state or replacement.
Medium	Bugs that can break the intended contract logic or expose it to DoS attacks, but do not cause direct loss funds.
Informational	Bugs that do not have a significant immediate impact and could be easily fixed.

Based on the feedback received from the Customer regarding the list of findings discovered by the Contractor, they are assigned the following statuses:

Status	Description
Fixed	Recommended fixes have been made to the project code and no longer affect its security.
Acknowledged	The Customer is aware of the finding. Recommendations for the finding are planned to be resolved in the future.

3. Summary of findings

Severity	# of Findings
Critical	0
High	8
Medium	3
Informational	42

4. Conclusion

Commits with all fixes: [683a355de3317278f5f09dcd8aa136e1a8f80639](#), [0d09e21c818542d6705b8a84a3233a473ac5fff3](#)
6 high and 3 medium and 42 informational severity issue was found, 5 out of 6 high, 2 out of 3 medium and 26 out of 42 informational severity issues were fixed.

Deployment

File name	Contract deployed on mainnet
ArrakisV2.sol	OxAfOf96e8702cB1b8160e43c8c020C608cD7B134d
ArrakisV2Beacon.sol	Ox1D91F6D917ec51dE53A5789c34fFF777a58759B6
ArrakisV2Factory.sol	OxECb8Ffcb2369EF188A082a662F496126f66c8288
ArrakisV2Helper.sol	OxO7d2CeB4869DFE17e8D48c92A71eDC3AE564449f
ArrakisV2Resolver.sol	Oxb11bb8ad710579Cc5ED16b1C8587808109c1f193
Pool.sol	Ox4cD412O4AA4C7438374256bD7bE85OeF9fcFaB84
Position.sol	OxF7cB77C8dCB22A1bb4435932f3515319721Faf44
Underlying.sol	Ox92CB4F7e4CB623E73D5Ec84A43669ADc757C2bd2

File name	Contract deployed on mainnet
PALMManager.sol	OxOa7D53FF9C56a3bD6A4A369f14ba3Ba523B3O13E
PALMTerms.sol	OxBO41f628e961598af9874BCf3OCC865f67fad3EE

5. Findings report



High

Possible loss of funds by the manager	Acknowledged
<p>Description</p> <p>The manager has full access to vault rebalance parameters. E.g. In the function <code>_rebalance</code> manager can:</p> <ol style="list-style-type: none">1. Burn all the liquidity in the uniswap pool (contract approves all its balance for transfer regardless of swap amount);2. Asset swap via low-level call;3. Pass his address as a recipient to the swap function or sandwich himself (or by someone else). <p>Worth noting that if vault uses the same uniswap pool for swaps then the sandwich attack becomes even cheaper (easier to skew the price because of the burning [1]).</p> <p>Recommendation</p> <p>We recommend calling swap routers via an interface (or implementing a new contract to manage swaps) to check all necessary input parameters and avoid making arbitrary low-level calls. Another way to address this issue may be adding restrains to the <code>SwapPayload::expectedMinReturn</code> parameter, but it may require an external oracle.</p> <p>Client's comments</p> <p>Left unresolved. From perspective of v2-core manager is intentionally a trusted party that passes potentially sensitive/manipulable parameters. For publicly accessible vaults, specific manager contract implementations will be used to yield more trustless manager</p>	
Possible uniswap pool manipulation	Fixed at 23ac78
<p>Description</p> <p>If, when calling the <code>burn</code> function, the vault doesn't have enough of either token0 or token1 (e.g. after the vault mints positions) it will <code>burn</code> some of the liquidity according to user input. Arbitrary users can, intentionally or not, burn all the pool's liquidity and the vault won't receive any yield until the next rebalance.</p> <p>On the other side, it will negatively affect all the uniswap users, if the vault holds a significant part of the pool's liquidity, e.g. using ArrakisV2 as a tool to aid in the sandwich attacks.</p> <p>Recommendation</p> <p>We recommend decreasing users' exposure to position burn functionality.</p>	

Possible fee hijacking (or DOS) by the manager	Fixed at 50706e
<p>Description</p> <p>Currently, Arrakis relies on the manager's good faith, since the vault quarries the manager's smart contract to get the <code>managerFeeBPS</code> parameter. Although the average scenario manager would be an instance of <code>PALMManager.sol</code> (<code>managerFeeBPS</code> is immutable), an owner can change the manager address. An attacker with manager access could set <code>managerFeeBPS = 10000 - arrakisFeeBPS</code> (or set <code>managerFeeBPS > 10000 - arrakisFeeBPS`</code> to break the vault) to steal all the yield (+ all the accrued manager fees).</p> <p>Recommendation</p> <p>We recommend storing the <code>managerFeeBPS</code> parameter value in the storage of <code>ArrakisV2Storage</code> contract with additional value restrains in setters (explicit max & min values) and limiting <code>managerFeeBPS <= hundredPercent</code> on setting.</p>	
Liquidity withdrawal from Uni position to Vault	Fixed at fecOcb
<p>Description</p> <p>Malicious users are able to move liquidity from uni to Arrakis vault, because the <code>burn</code> function withdraws liquidity from Uni positions proportionally to <code>burnAmount/totalSupply</code> ArrakisV2.sol#L145-L154</p> <ol style="list-style-type: none"> 1. Take a flashloan 2. Mint huge amount of shares 3. Burn them <p>Recommendation</p> <p>We recommend removing buffer</p>	
First minter can change LP token pricing	Fixed at 23ac78
<p>Description</p> <p>At the lines ArrakisV2.sol#L83-L84:</p> <pre>amount0 = FullMath.mulDivRoundingUp(mintAmount_, current0, denominator); amount1 = FullMath.mulDivRoundingUp(mintAmount_, current1, denominator);</pre> <p>When <code>totalSupply</code> is 0, the variables <code>current0</code>, <code>current1</code>, <code>denominator</code> are set to <code>init0</code>, <code>init1</code>, 1 ether respectively. If <code>init0</code> and <code>init1</code> are less than 1e18, then the first minter can mint <code>mintAmount_ = 1 wei</code> for <code>amount0 = 1 wei</code> and <code>amount1 = 1 wei</code> regardless of the values <code>init0</code> and <code>init1</code>.</p> <p>Recommendation</p> <p>It is recommended to ensure <code>amount0</code>, <code>amount1</code> are proportional to <code>init0</code>, <code>init1</code>.</p>	
Vault can renew the term for free	Fixed at 465466
<p>Description</p> <p>The vault owner mints 1 wei when deploying a vault with <code>openTerm()</code>. Then they can renew term for free since in the function <code>renewTerm()</code> at the line PALMTerms.sol#L153, <code>emolumentShares</code> are calculated from the balance of <code>PALMTerms</code> and the function <code>increaseLiquidity()</code> doesn't mint new LP tokens.</p> <p>Recommendation</p> <p>It is recommended to mint new LP tokens in <code>increaseLiquidity()</code>.</p>	

Vault owner can lose tokens when increasing liquidity	Fixed at 9Oe5c1
<p>Description</p> <p>If the vault owner burns all LP tokens owned by <code>PALMTerms</code> with <code>decreaseLiquidity()</code>, then any future call to <code>increaseLiquidity()</code> would lose tokens in the vault's address. PALMTerms.sol#L170</p> <p>Recommendation</p> <p>It is recommended to mint new LP tokens in <code>increaseLiquidity()</code>.</p>	

Rebalance DoS	Fixed at fecOcb
<p>Description</p> <p>Attacker can block <code>rebalance</code> minting some dust to Arrakis Uni position before <code>rebalance</code> is executed, trx will be reverted because ArrakisV2.sol#L231 Position.sol#L24 <code>liquidity</code> won't be zero.</p> <p>Recommendation</p> <p>We recommend adding and removing ranges when they get created and get drained.</p>	

Medium

Checking caller	Fixed at 465466
<p>Description</p> <p>Function addVault() doesn't check <code>msg.sender</code>. Anyone can call this function and add vault if its owner is <code>terms</code>. This can lead to DOS of contract and unintended behavior.</p> <p>Recommendation</p> <p>It is recommended to add <code>msg.sender</code> check, if it is vault's owner or <code>terms</code>.</p>	

No check if liquidity > 0 in standardBurnParams()	Fixed at 50706e
<p>Description</p> <p>In the function <code>standardBurnParams()</code> at the lines ArrakisV2Resolver.sol#L220-L225:</p> <pre>burns[i] = BurnLiquidity({ liquidity: SafeCast.toUint128(FullMath.mulDiv(liquidity, amountToBurn_, totalSupply)), range: ranges[i] });</pre> <p>There is no check if <code>burns[i].liquidity > 0</code>. If <code>burns[i].liquidity = 0</code>, then <code>burn()</code> in <code>ArrakisV2</code> will revert.</p> <p>Recommendation</p> <p>It is recommended to check if <code>burns[i].liquidity > 0</code> and not include range if <code>liquidity = 0</code>.</p>	

Anyone can call <code>renewTerm()</code>	Acknowledged
<p>Description</p> <p>In the function <code>renewTerm()</code> at the line PALMTerms.sol#L143, there is no check if the caller is the owner of the vault. An attacker can frontrun <code>closeTerm()</code> and call <code>renewTerm()</code>, then the vault would pay the emolument twice.</p> <p>Recommendation</p> <p>It is recommended to add the modifier <code>requireIsOwner()</code>.</p> <p>Client's comments</p> <div> <p>By design. RenewTerm can be called only when term management time ends. If client want to terminate the term with palm, they can call closeTerm function before end of management time. If they do not, renewal is automatic and we (or anyone) will promptly call renewTerm to extract the fee of the last epoch.</p> </div>	

Informational

Redundant modifier	Fixed at 9Oe5c1
<p>Description</p> <p><code>PALMTerms</code> deploys new vaults with himself as an owner via <code>openTerm</code> function, as well <code>PALMManager</code>'s <code>addVault</code> function accepts only <code>term-owned</code> vaults.</p> <p><code>PALMManager</code>'s <code>removeVault</code>, <code>setVaultData</code>, <code>setVaultStraByName</code> and <code>withdrawVaultBalance</code> methods have modifier <code>onlyVaultOwner</code>, which is redundant as these methods are always called from <code>PALMTerms</code>. <code>addVault</code> has the <code>onlyPALMTermsVaults</code> modifier (you can add random garbage to the mapping by bypassing it), which could be replaced with <code>onlyPALMTerms</code> modifier too, since it is always supposed to be called from <code>PALMTerms</code>.</p> <p>Recommendation</p> <p>We recommend replacing <code>onlyVaultOwner</code> and <code>onlyPALMTermsVaults</code> modifiers with <code>onlyPALMTerms</code>.</p>	
Unused modifier code duplication	Fixed at 9Oe5c1
<p>Description</p> <p><code>PALMManagerStorage</code> <code>removeVault</code> function has the same <code>require</code> check as <code>onlyManagedVaults</code> modifier.</p> <p>Recommendation</p> <p>We recommend replacing the duplicated line with <code>onlyManagedVaults</code> modifier.</p>	
Reducing SLOAD operations	Fixed at 23ac78
<p>Description</p> <p>You can reduce SLOAD operations, hence reducing gas spending, by copying often-used storage variables to the memory.</p> <p>Recommendation</p> <p>We recommend copying storage variables to memory in case of multiple variable read operations. For example, <code>ArrakisV2::_rebalance</code> function could be optimized by copying <code>factory</code>, <code>token0</code> and <code>token1</code> into the memory before all loops.</p>	

Event indexed fields	Fixed at 23ac78
<p>Description</p> <p>In ArrakisV2Storage, events LogMint, LogBurn and LPBurned could be modified to have reciever (user) field indexed to improve parsing user balance.</p> <p>Recommendation</p> <p>We recommend making reciever(user) field indexed.</p>	
Typo in function naming	Fixed at 9Oe5c1
<p>Description</p> <p>PALMManagerStorage function setVaultStraByName should be spelled setVaultStratByName.</p> <p>Recommendation</p> <p>We recommend changing the naming and all its references in IPALMManager and PALMTermsStorage.</p>	
Direct token transfers	Acknowledged
<p>Description</p> <p>If ArrakisV2 contract has pool tokens on its balance (excluding arraking and manager balances) and if its total supply is zero then anyone who calls mint() function will get more tokens than he transferred. It is possible if minting LP tokens and transferring tokens to vault take place not atomically and mint is not restricted.</p> <p>Recommendation</p> <p>It is recommended to avoid direct transfers to ArrakisV2 vault if mint is not restricted.</p>	
Range existence check	Fixed at 23ac78
<p>Description</p> <p>In cycle at Line 169 burns_[i].range may not be present in the ranges array.</p> <p>Same issue:</p> <ul style="list-style-type: none">• In cycle at Line 322 <p>Recommendation</p> <p>It is recommended to add require statement if range exists.</p>	
Gas optimisation in range deletion	Acknowledged
<p>Description</p> <p>During the deletion of range at Line 269 all elements after it are moved left in cycle at Lines 271-273.</p> <p>Recommendation</p> <p>It is recommended to swap elements to delete and the last element of the array and call pop() function.</p>	

Zero address check	Fixed at 23ac78
--------------------	---------------------------------

Description

At [Line 144](#) `params.owner` variable is not checked for zero address. If ownership is transferred to zero address, admin functionality will be unavailable and it can not be restored.

Same issue:

- [Line 43](#)
- [Line 49](#)
- [Line 13](#)
- [Line 136](#)
- [Line 88](#)

Recommendation

It is recommended to add `require` statement to check variables and parameters for zero address.

Bad readability	Fixed at 23ac78
-----------------	---------------------------------

Description

At [Line 149](#) variables `init0` and `init1` are set during event emitting.

Same issue:

- [Line 161](#)
- [Line 193](#)
- [Line 197](#)
- [Line 101](#)
- [Line 113](#)
- [Line 125](#)
- [Line 135](#)
- [Line 215](#)
- [Line 293](#)

Recommendation

It is recommended to separate variable initialization and event emitting.

Redundant check	Acknowledged
-----------------	--------------

Description

At [Line 171](#) `_pools.contains()` is redundant because it is checked in `_pools.remove()`. Same comment with `_pool.add()`.

Same issue:

- [Line 185](#)
- [Line 234](#)
- [Line 248](#)
- [Line 312](#)
- [Line 261](#)

Recommendation

It is recommended to use `require` statement with `remove()` and `add()` EnumerableSet' methods instead of `contains()` method.

Gas optimisation in vaults()	Acknowledged
<p>Description</p> <p>Function <code>vaults()</code> iterates through <code>_vaults</code> enumerable set and saves values to an allocated memory array. The enumerable set has built-in function <code>values()</code> for that purpose.</p> <p>Recommendation</p> <p>It is recommended to use the enumerable set's function <code>values</code> to reduce code size and gas costs.</p>	
Gas optimisation in conversion from int to string	Acknowledged
<p>Description</p> <p>Function <code>_uint2str</code> is more gas expansive than Openzeppelin function <code>toString()</code>.</p> <p>Recommendation</p> <p>It is recommended to use Openzeppelin function <code>toString()</code> to convert <code>uint</code> to <code>str</code>.</p>	
Code refactoring	Acknowledged
<p>Description</p> <p>In the function <code>standartBurnParams()</code> helper's function <code>totalUnderlyingWithFees()</code> is called and then leftovers are calculated. <code>ArrakisV2Helper</code> contract has a function that calculates leftovers - <code>totalUnderlyingWithFeesAndLeftOver()</code>.</p> <p>Recommendation</p> <p>It is recommended to use <code>ArrakisV2Helper</code>'s function <code>totalUnderlyingWithFeesAndLeftOver()</code> to get all necessary data.</p>	
Zero liquidity check	Fixed at 23ac78
<p>Description</p> <p>At Line 205 liquidity of position is returned and saved to <code>burns</code> array. If liquidity is zero, it is redundant to add position info to an array.</p> <p>Recommendation</p> <p>It is recommended to add <code>require</code> statement to check if the position has liquidity.</p>	
Blacklisting strategy	Acknowledged
<p>Description</p> <p>In <code>PALMManagerStorage</code> contract there is a method to whitelist strategies, but there is no method to blacklist them if a strategy is irrelevant or ineffective.</p> <p>Recommendation</p> <p>It is recommended to add a method to blacklist strategies.</p>	

Redundant approve call	Acknowledged
<p>Description</p> <p>At Lines 104-105 vault's allowance is set to zero for both tokens, but it is already zero.</p> <p>Recommendation</p> <p>It is recommended to remove <code>safeApprove()</code> calls setting allowance to zero.</p>	
Unmatched to documentation	Acknowledged
<p>Description</p> <p>Function increaseLiquidity() simply transfers tokens from <code>msg.sender</code> to vaults. Based on docs, it should call <code>mint()</code> function.</p> <p>Same issue:</p> <ul style="list-style-type: none"> In function rebalance() during deposits there is no check for current and average price. <p>Recommendation</p> <p>It is recommended to leave a comment if it is intended behaviour.</p>	
No max limit for ranges	Acknowledged
<p>Description</p> <p>In the function <code>rebalance()</code> at the line ArrakisV2.sol#L241.</p> <p>The manager can add ranges, but there is no max limit for the number of ranges. If the <code>ranges</code> array is too big, it will be impossible to mint and burn LP tokens since there will not be enough gas in the block.</p> <p>Recommendation</p> <p>It is recommended to limit the maximum number of ranges.</p>	
Variable totalSupply shadows function	Fixed at 23ac78
<p>Description</p> <p>At the lines ArrakisV2.sol#L63, ArrakisV2.sol#L106:</p> <pre>uint256 totalSupply = totalSupply();</pre> <p>The variable <code>totalSupply</code> shadows function <code>totalSupply()</code>.</p> <p>Recommendation</p> <p>It is recommended to rename the variable.</p>	

Possible to use LP token as vault token	Acknowledged
---	--------------

Description

In the function `mint()` at the lines [ArrakisV2.sol#L86-L94](#):

```
_mint(receiver_, mintAmount_);
```

```
// transfer amounts owed to contract
if (amount0 > 0) {
    token0.safeTransferFrom(msg.sender, me, amount0);
}
if (amount1 > 0) {
    token1.safeTransferFrom(msg.sender, me, amount1);
}
```

The LP token is minted before `token0` and `token1` are pulled to the contract, which means that it is possible to use LP token of the vault as `token0` or `token1`.

Recommendation

It is recommended to mint LP tokens after pulling `token0` and `token1`.

Possible to burn zero LP tokens	Fixed at 23ac78
---------------------------------	---------------------------------

Description

In the function `burn()` at the line [ArrakisV2.sol#L101](#), the parameter `burnAmount_` is not checked to be bigger than `0`.

Recommendation

It is recommended to check if `burnAmount_ > 0`.

Incorrect event emit	Acknowledged
----------------------	--------------

Description

In the function `burn()` at the line [ArrakisV2.sol#L216](#):

```
emit LogUncollectedFees(underlying.fee0, underlying.fee1);
```

Some fees will be collected when burning liquidity, so this event emit `LogUncollectedFees()` is incorrect.

Recommendation

It is recommended to emit `LogUncollectedFees()` before the `return` statement at the line [ArrakisV2.sol#L159](#).

TODO comments	Fixed at 23ac78
---------------	---------------------------------

Description

At the lines:

- [ArrakisV2.sol#L238](#)
- [ArrakisV2Resolver.sol#L111](#)

`TODO` comments should be removed before deployment.

Recommendation

It is recommended to remove `TODO` comments.

Using a number literal	Acknowledged
<p>Description</p> <ul style="list-style-type: none"> • In the function <code>_applyFees()</code> at the lines ArrakisV2.sol#L462-L45 • At the lines ArrakisV2Resolver.sol#L133-L134 • At the line ArrakisV2Resolver.sol#L292 • At the line PALMTermsStorage.sol#L87 <p>The literal <code>10000</code> can be replaced with a constant variable for better readability.</p> <p>Recommendation</p> <p>It is recommended to use a constant variable instead of a literal.</p>	

Inadequate view functions in ArrakisV2Storage	Acknowledged
<p>Description</p> <p>At the lines ArrakisV2Storage.sol#L65-L66:</p> <pre>EnumerableSet.AddressSet internal _pools; EnumerableSet.AddressSet internal _routers;</pre> <p>The are no view functions for pools and routers used in the vault.</p> <p>Recommendation</p> <p>It is recommended to add view functions for pools and routers.</p>	

Function removePools() doesn't remove ranges	Acknowledged
<p>Description</p> <p>In the function <code>removePools()</code> at the lines ArrakisV2Storage.sol#L169-L176:</p> <pre>function removePools(address[] calldata pools_) external onlyOwner { for (uint256 i = 0; i < pools_.length; i++) { require(_pools.contains(pools_[i]), "NP"); _pools.remove(pools_[i]); } emit LogRemovePools(pools_); }</pre> <p>The pool is only removed from <code>_pools</code>, but there might be active positions with same fee tier in <code>_ranges</code>.</p> <p>Recommendation</p> <p>It is recommended to remove ranges with the same fee tier when removing a pool and ensure they have no liquidity.</p>	

Functions can be declared as external	Fixed at 23ac78
<p>Description</p> <ul style="list-style-type: none"> The function <code>vaults()</code> at the line ArrakisV2Factory.sol#L57 can be <code>external</code> since it is not used internally. The function <code>getProxyAdmin()</code> at the line ArrakisV2FactoryStorage.sol#L86 can be <code>external</code> since it is not used internally. The function <code>getProxyImplementation()</code> at the line ArrakisV2FactoryStorage.sol#L96 can be <code>external</code> since it is not used internally. The function <code>getAmountsForLiquidity()</code> at the line ArrakisV2Resolver.sol#L266 can be <code>external</code> since it is not used internally. <p>Recommendation</p> <p>It is recommended to change these functions to <code>external</code>.</p>	

Function vaults() may run out of gas	Fixed at 23ac78
<p>Description</p> <p>The function <code>vaults()</code> at the lines ArrakisV2Factory.sol#L57-L65:</p> <pre>function vaults() public view returns (address[] memory) { uint256 length = numVaults(); address[] memory vs = new address[](length); for (uint256 i = 0; i < length; i++) { vs[i] = _vaults.at(i); } return vs; }</pre> <p>If there are a lot of vaults deployed, this function will be unusable since there will not be enough gas in the block to loop over all of the vaults.</p> <p>Recommendation</p> <p>It is recommended to change the function to get one vault with parameter <code>index</code>.</p>	

Unnecessary external calls	Fixed at 23ac78
<p>Description</p> <p>At the lines:</p> <ul style="list-style-type: none"> ArrakisV2Helper.sol#L36 ArrakisV2Helper.sol#L71 ArrakisV2Helper.sol#L88 ArrakisV2Resolver.sol#L78 ArrakisV2Resolver.sol#L122 ArrakisV2Resolver.sol#L206 <p>The external calls <code>vault_.factory()</code> are unnecessary since <code>factory</code> is already available as an immutable variable.</p> <p>Recommendation</p> <p>It is recommended to use variable <code>factory</code> instead of an external call.</p>	

Inefficient function ranges() in ArrakisV2Helper	Fixed at 23ac78
<p>Description</p> <p>At the line ArrakisV2Helper.sol#L171, the function <code>ranges()</code> is gas inefficient. It can be replaced with a getter function in <code>ArrakisV2Storage</code> that would return <code>ranges</code>.</p> <p>Recommendation</p> <p>It is recommended to add a getter function in <code>ArrakisV2Storage</code> to return <code>ranges</code>.</p>	
Array operators can be changed to EnumerableSet.AddressSet	Fixed at 9Oe5c1
<p>Description</p> <p>At the line PALMManagerStorage.sol#L65.</p> <p>The array <code>operators</code> has costly operations when adding, removing operators and checking if an operator exists. It can be optimized if <code>operators</code> is a <code>EnumerableSet.AddressSet</code>.</p> <p>Recommendation</p> <p>It is recommended to change <code>operators</code> to a <code>EnumerableSet.AddressSet</code>.</p>	
gelatoFeeCollector can be set to address(0)	Fixed at 9Oe5c1
<p>Description</p> <p>At the lines PALMManagerStorage.sol#L138, PALMManagerStorage.sol#L217.</p> <p>The variable <code>gelatoFeeCollector</code> can be set to <code>address(0)</code>.</p> <p>Recommendation</p> <p>It is recommended to ensure <code>gelatoFeeCollector</code> can never be <code>address(0)</code>.</p>	
Can fund vault balance with msg.value = 0	Fixed at 9Oe5c1
<p>Description</p> <p>In the function <code>fundVaultBalance()</code> at the lines PALMManagerStorage.sol#L275-L283, there is no check if <code>msg.value > 0</code>.</p> <p>Recommendation</p> <p>It is recommended to check if <code>msg.value > 0</code>.</p>	
Irrelevant comments	Fixed at 9Oe5c1
<p>Description</p> <p>At the lines PALMTerms.sol#L83, PALMTerms.sol#L92, the comments are irrelevant.</p> <p>Recommendation</p> <p>It is recommended to move the comment at line PALMTerms.sol#L83 and remove the comment at line PALMTerms.sol#L92.</p>	

User can mint LP tokens to themselves in PALMTerms	Fixed at 9Oe5c1
<p>Description</p> <p>At the lines PALMTerms.sol#L93-L102, <code>token0</code> and <code>token1</code> are transferred to the contract before <code>vaultV2.setRestrictedMint(address(this));</code> is called. If <code>token0</code> or <code>token1</code> is an <code>ERC777</code> token, then the caller can mint LP tokens before the mint is restricted.</p> <p>Recommendation</p> <p>It is recommended to add restricted mint as a vault deployment parameter.</p>	
setRestrictedMint() can be frontrun	Acknowledged
<p>Description</p> <p>At the lines ArrakisV2Storage.sol#L196-L198, the function <code>setRestrictedMint()</code> can be frontrun to mint tokens before restricted mint is set since <code>restrictedMint = address(0)</code> by default.</p> <p>Recommendation</p> <p>It is recommended to add restricted mint as a vault deployment parameter.</p>	
Gas optimization in renewTerm()	Fixed at 9Oe5c1
<p>Description</p> <p>In the function <code>renewTerm()</code> at the line PALMTerms.sol#L149, it is possible to use memory variable <code>manager_</code> to save gas.</p> <p>Recommendation</p> <p>It is recommended to change to <code>manager_.renewTerm(address(vault_));</code></p>	
User can receive less than expected when decreasing liquidity	Fixed at 9Oe5c1
<p>Description</p> <p>In the function <code>decreaseLiquidity()</code> at the lines PALMTerms.sol#L190-L194.</p> <pre>require(amount0 >= decreaseBalance_.amount0Min && amount1 >= decreaseBalance_.amount1Min, "PALMTerms: received below minimum");</pre> <p>When calling <code>decreaseLiquidity()</code> the user expects to receive <code>decreaseBalance_.amount0Min</code>, but actually they receive <code>amount0 - emolumentAmt0</code> which might be smaller than <code>decreaseBalance_.amount0Min</code> since there is only a check for <code>amount0</code>.</p> <p>Recommendation</p> <p>It is recommended to check if <code>amount0 - emolumentAmt0 >= decreaseBalance_.amount0Min</code>.</p>	

Incorrect version	Fixed at 23ac78
<p>Description</p> <p>Incorrect version here, we think, that it is supposed to be 2.O. ArrakisV2FactoryStorage.sol#L31</p> <p>Recommendation</p> <p>We recommend changing the version to 2.O or another, different from 1.O, to not be confused with the first version before deploying.</p>	

Gas optimizations in for loop	Fixed at 23ac78
<p>Description</p> <p>There're some suboptimal maths here: Position.sol#L40 Underlying.sol#L38 ArrakisV2.sol#L254 ... ArrakisV2Resolver.sol#L288 PALMManagerStorage.sol#L227 ... PALMManagerStorage.sol#L436</p> <p>Recommendation</p> <p>We recommend replacing code like:</p> <pre>for (uint i = 0; i < length; i++) { ... }</pre> <p>to:</p> <pre>for (uint i; i < length; ++i) { ... }</pre> <p>Cause it will save some computed units.</p>	

Code duplicating	Fixed at 9Oe5c1
<p>Description</p> <p>Here, we set <code>vaults[vault_].balance</code> to zero, but deleting this element makes the same. PALMManagerStorage.sol#L371.</p> <p>Recommendation</p> <p>We recommend deleting duplicated functionality.</p>	

7. Appendix B. Slither



Informational/High/low-level-calls

Low level call in [ArrakisV2FactoryStorage.getProxyImplementation\(address\)](#): - [\(success, returndata\) = proxy.staticcall\(Ox5c6Oda1b\)](#).

Low level call in [ArrakisV2FactoryStorage.getProxyAdmin\(address\)](#): - [\(success, returndata\) = proxy.staticcall\(Oxf851a44O\)](#).

Low level call in [ArrakisV2._rebalance\(Rebalance\)](#): - [\(success\) = rebalanceParams._swap.router.call\(rebalanceParams._swap.payload\)](#).

Informational/High/naming-convention

Parameter [ArrakisV2FactoryStorage.initialize\(address\).owner](#) is not in mixedCase

Variable [ArrakisV2Storage._pools](#) is not in mixedCase

Constant [ArrakisV2FactoryStorage.version](#) is not in UPPER_CASE_WITH_UNDERSCORES

Constant [ArrakisV2Storage.arrakisFeeBPS](#) is not in UPPER_CASE_WITH_UNDERSCORES

Variable [ArrakisV2FactoryStorage._vaults](#) is not in mixedCase

Variable [ArrakisV2Storage._routers](#) is not in mixedCase

Informational/Medium/costly-loop

[ArrakisV2.rebalance\(Range\[\],Rebalance,Range\[\]\)](#) has costly operations inside a loop: - [delete ranges\[index\]](#)

[ArrakisV2.rebalance\(Range\[\],Rebalance,Range\[\]\)](#) has costly operations inside a loop: - [ranges.pop\(\)](#)

Informational/Medium/similar-names

Variable [Underlying.getUnderlyingBalances\(PositionUnderlying\).tokensOwedO](#) is too similar to [Underlying.getUnderlyingBalances\(PositionUnderlying\).tokensOwed1](#)

Variable [ArrakisV2.uniswapV3MintCallback\(uint256,uint256,bytes\).amountOOwed_](#) is too similar to [ArrakisV2.uniswapV3MintCallback\(uint256,uint256,bytes\).amount1Owed_](#)

Variable [IArrakisV2Resolver.getMintAmounts\(IArrakisV2,uint256,uint256\).amountOMax_](#) is too similar to [ArrakisV2Resolver.getMintAmounts\(IArrakisV2,uint256,uint256\).amount1Max_](#)

Variable [ArrakisV2._rebalance\(Rebalance\).balanceOAfter](#) is too similar to [ArrakisV2._rebalance\(Rebalance\).balance1After](#)

Variable [ArrakisV2Resolver.getMintAmounts\(IArrakisV2,uint256,uint256\).amountOMax_](#) is too similar to [IArrakisV2Resolver.getMintAmounts\(IArrakisV2,uint256,uint256\).amount1Max_](#)

Variable [Underlying._getFeesEarned\(GetFeesPayload\).feeGrowthOutsideOLower](#) is too similar to [Underlying._getFeesEarned\(GetFeesPayload\).feeGrowthOutside1Lower](#)

Variable [ArrakisV2._rebalance\(Rebalance\).aggregatorO](#) is too similar to [ArrakisV2._rebalance\(Rebalance\).aggregator1](#)

Variable [Underlying._getFeesEarned\(GetFeesPayload\).feeGrowthOutsideOUpper](#) is too similar to [Underlying._getFeesEarned\(GetFeesPayload\).feeGrowthOutside1Upper](#)

Variable `Underlying.computeMintAmounts(uint256,uint256,uint256,uint256,uint256).amountOMax` is too similar to `Underlying.computeMintAmounts(uint256,uint256,uint256,uint256,uint256).amount1Max`

Variable [ArrakisV2Resolver.getMintAmounts\(IArrakisV2,uint256,uint256\).amountOMax](#) is too similar to [ArrakisV2Resolver.getMintAmounts\(IArrakisV2,uint256,uint256\).amount1Max](#)

Variable `Underlying.computeMintAmounts(uint256,uint256,uint256,uint256,uint256).amountOMint` is too similar to `Underlying.computeMintAmounts(uint256,uint256,uint256,uint256,uint256).amount1Mint`

Variable [Underlying.getUnderlyingBalances\(PositionUnderlying\).amountOCurrent](#) is too similar to [Underlying.getUnderlyingBalances\(PositionUnderlying\).amount1Current](#)

Variable [ArrakisV2Storage.managerBalance0](#) is too similar to [ArrakisV2Storage.managerBalance1](#)

Variable `IArrakisV2Resolver.getMintAmounts(IArrakisV2,uint256,uint256).amountOMax_` is too similar to `IArrakisV2Resolver.getMintAmounts(IArrakisV2,uint256,uint256).amount1Max_`

Variable [Underlying.getUnderlyingBalances\(PositionUnderlying\).feeGrowthInsideOLast](#) is too similar to [Underlying.getUnderlyingBalances\(PositionUnderlying\).feeGrowthInside1Last](#)

Variable [ArrakisV2Storage.arrakisBalance0](#) is too similar to [ArrakisV2Storage.arrakisBalance1](#)

Variable [ArrakisV2Storage.addPools\(uint24\[\],address,address\).tokenOAddr](#) is too similar to [ArrakisV2Storage.addPools\(uint24\[\],address,address\).token1Addr](#)

Variable [ArrakisV2._rebalance\(Rebalance\).balanceOBefore](#) is too similar to [ArrakisV2._rebalance\(Rebalance\).balance1Before](#)

Informational/Medium/too-many-digits

Underlying computeFeesEarned(ComputeFeesPayload) uses literals with too many digits: - fee = FullMath.mulDiv(computeFees_.liquidity,feeGrowthInside - computeFees_.feeGrowthInsideLast,Ox1000).

Low/High/variable-scope

Variable '[ArrakisV2Factory._preDeploy\(InitializePayload,bool\).result](#)' in [ArrakisV2Factory._preDeploy\(InitializePayload,bool\)](#) potentially used before declaration: [name = result](#)

Variable '[ArrakisV2.rebalance\(Range\[\],Rebalance,Range\[\]\).exist](#)' in [ArrakisV2.rebalance\(Range\[\],Rebalance,Range\[\]\)](#) potentially used before declaration: [\(exist,index\) = Position.rangeExist\(ranges,rangesToRemove_\[i_scope_O\]\)](#).

Variable '[Manager.getManagerFeeBPS\(IManager\).feeBPS](#)' in [Manager.getManagerFeeBPS\(IManager\)](#) potentially used before declaration: [feeBPS](#)

Low/Medium/calls-loop

[Underlying._getFeesEarned\(GetFeesPayload\)](#) has external calls inside a loop: [\(feeGrowthOutsideOLower,feeGrowthOutside1Lower\) = feeInfo_.pool.ticks\(feeInfo_.lowerTick\)](#).

[ArrakisV2Resolver.standardRebalance\(RangeWeight\[\],IArrakisV2\)](#) has external calls inside a loop: [\(sqrtPriceX96\) = IUniswapV3Pool\(vaultV2_.factory\(\).getPool\(tokenOAddr,token1Addr,rangeWeight.range.feeTier\)\).slotO\(\)](#).

[ArrakisV2Resolver.standardBurnParams\(uint256,IArrakisV2\)](#) has external calls inside a loop: [\(liquidity,None,None,None,None\) = IUniswapV3Pool\(vaultV2_.factory\(\).getPool\(address\(vaultV2_.tokenO\(\)\),address\(vaultV2_.token1\(\)\),ranges\[i\].feeTier\)\).positions\(Position.getPositionId\(address\(vaultV2_\),ranges\[i\].lowerTick,ranges\[i\].upperTick\)\)](#).

[ArrakisV2FactoryStorage.upgradeVaults\(address\[\]\)](#) has external calls inside a loop: [ITransparentUpgradeableProxy\(vaults_\[i\]\).upgradeTo\(arrakisV2Beacon.implementation\(\)\)](#).

[ArrakisV2Helper._getAmountsAndFeesFromLiquidity\(address,address,Range,address\)](#) has external calls inside a loop: [pool = IUniswapV3Pool\(factory.getPool\(tokenO_,token1_,range_.feeTier\)\)](#).

[Underlying.underlying\(RangeData\)](#) has external calls inside a loop: [\(sqrtPriceX96,tick\) = underlying_.pool.slotO\(\)](#).

[ArrakisV2Helper.ranges\(IArrakisV2\)](#) has external calls inside a loop: [rgs\[i\] = vault_.ranges\(i\)](#).

[ArrakisV2Helper.ranges\(IArrakisV2\)](#) has external calls inside a loop: [vault_.ranges\(index\)](#).

[ArrakisV2Resolver.standardRebalance\(RangeWeight\[\],IArrakisV2\)](#) has external calls inside a loop: [\(liquidity,None,None,None,None\) = IUniswapV3Pool\(vaultV2_.factory\(\).getPool\(tokenOAddr,token1Addr,ranges\[i\].feeTier\)\).positions\(Position.getPositionId\(address\(vaultV2_\),ranges\[i\].lowerTick,ranges\[i\].upperTick\)\)](#).

[Underlying._getFeesEarned\(GetFeesPayload\)](#) has external calls inside a loop: [payload = ComputeFeesPayload\(feeInfo_.feeGrowthInsideOLast,feeGrowthOutsideOLower,feeGrowthOutsideOUpper,feeInfo_.pool.feeGrowthGlobalOX128\(\),feeInfo_.pool,feeInfo_.liquidity,feeInfo_.tick,feeInfo_.lowerTick,feeInfo_.upperTick\)](#).

[Underlying._getFeesEarned\(GetFeesPayload\)](#) has external calls inside a loop: [\(feeGrowthOutsideOUpper,feeGrowthOutside1Upper\) = feeInfo_.pool.ticks\(feeInfo_.upperTick\)](#).

[ArrakisV2._withdraw\(IUniswapV3Pool,int24,int24,uint128\)](#) has external calls inside a loop: [\(withdraw.burnO,withdraw.burn1\) = pool_.burn\(lowerTick_,upperTick_,liquidity_\)](#).

[Underlying.getUnderlyingBalances\(PositionUnderlying\)](#) has external calls inside a loop: [\(liquidity,feeGrowthInsideOLast,feeGrowthInside1Last,tokensOwedO,tokensOwed1\) = positionUnderlying_.pool.positions\(positionUnderlying_.positionId\)](#).

[ArrakisV2.rebalance\(Range\[\],Rebalance,Range\[\]\)](#) has external calls inside a loop: [pool = factory.getPool\(address\(tokenO\),address\(token1\),ranges_\[i\].feeTier\)](#).

[ArrakisV2._withdraw\(IUniswapV3Pool,int24,int24,uint128\)](#) has external calls inside a loop: [\(collectO,collect1\) = pool_.collect\(address\(this\),lowerTick_,upperTick_,type\(\)\(uint128\).max,type\(\)\(uint128\).max\)](#).

[ArrakisV2FactoryStorage.makeVaultsImmutable\(address\[\]\)](#) has external calls inside a loop: [ITransparentUpgradeableProxy\(vaults_\[i\]\).changeAdmin\(address\(1\)\)](#).

[Underlying._getFeesEarned\(GetFeesPayload\)](#) has external calls inside a loop: [payload.feeGrowthGlobal = feeInfo_.pool.feeGrowthGlobal1X128\(\)](#).

[Underlying.totalUnderlyingWithFees\(UnderlyingPayload\)](#) has external calls inside a loop: [pool = IUniswapV3Pool\(underlyingPayload_.factory.getPool\(underlyingPayload_.tokenO,underlyingPayload_.token1,underlyingPayload_.ranges\[i\].feeTier\)\)](#).

[ArrakisV2.burn\(BurnLiquidity\[\],uint256,address\)](#) has external calls inside a loop: [withdraw = withdraw\(IUniswapV3Pool\(factory.getPool\(address\(tokenO\),address\(token1\),burns\[i\].range.feeTier\)\),burns_\[i\].range.lowerTick,burns_\[i\].range.upperTick,burns_\[i\].liquidity\)](#).

[ArrakisV2FactoryStorage.upgradeVaultsAndCall\(address\[\],bytes\[\]\)](#) has external calls inside a loop: [ITransparentUpgradeableProxy\(vaults_\[i\]\).upgradeToAndCall\(arrakisV2Beacon.implementation\(\),datas_\[i\]\)](#).

Low/Medium/missing-zero-check

[ArrakisV2FactoryStorage.getProxyImplementation\(address\).proxy](#) lacks a zero-check on : - [\(success,returndata\) = proxy.staticcall\(0x5c60da1b\)](#).

[ArrakisV2Storage.setRestrictedMint\(address\).minter_](#) lacks a zero-check on : - [LogRestrictedMint\(restrictedMint = minter_\)](#).

[ArrakisV2FactoryStorage.getProxyAdmin\(address\).proxy](#) lacks a zero-check on : - [\(success,returndata\) = proxy.staticcall\(0xf851a440\)](#).

Low/Medium/reentrancy-events

Reentrancy in [ArrakisV2.withdrawManagerBalance\(\)](#): External calls: - [tokenO.safeTransfer\(address\(manager\),amountO\)](#) - [token1.safeTransfer\(address\(manager\),amount1\)](#). Event emitted after the call(s): - [LogWithdrawManagerBalance\(amountO,amount1\)](#).

Reentrancy in [ArrakisV2._rebalance\(Rebalance\)](#): External calls: - [tokenO.safeApprove\(address\(rebalanceParams_.swap.router\),O\)](#) -

[token1.safeApprove\(address\(rebalanceParams_.swap.router\),O\)](#) -
[tokenO.safeApprove\(address\(rebalanceParams_.swap.router\),balanceOBefore\)](#) -
[token1.safeApprove\(address\(rebalanceParams_.swap.router\),balance1Before\)](#) - [\(success\) =](#)
[rebalanceParams_.swap.router.call\(rebalanceParams_.swap.payload\)](#) Event emitted after the call(s): -
[LogRebalance\(rebalanceParams_\)](#)

Reentrancy in [ArrakisV2.mint\(uint256,address\)](#): External calls: -
[tokenO.safeTransferFrom\(msg.sender,me,amountO\)](#) - [token1.safeTransferFrom\(msg.sender,me,amount1\)](#) Event
emitted after the call(s): - [LogMint\(receiver_,mintAmount_,amountO,amount1\)](#) - [LogUncollectedFees\(feeO,fee1\)](#)

Reentrancy in [ArrakisV2Factory.deployVault\(InitializePayload,bool\)](#): External calls: - [vault =](#)
[preDeploy\(params,isBeacon_\)](#) - [vault = address\(new BeaconProxy\(address\(arrakisV2Beacon\),data\)\)](#) - [vault =](#)
[address\(new TransparentUpgradeableProxy\(arrakisV2Beacon.implementation\(\),address\(this\),data\)\)](#) Event
emitted after the call(s): - [VaultCreated\(msg.sender,vault\)](#)

Reentrancy in [ArrakisV2.burn\(BurnLiquidity\[\],uint256,address\)](#): External calls: -
[tokenO.safeTransfer\(receiver_,amountO\)](#) - [token1.safeTransfer\(receiver_,amount1\)](#) Event emitted after the call(s):
- [LPBurned\(msg.sender,total.burnO,total.burn1\)](#) - [LogBurn\(receiver_,burnAmount_,amountO,amount1\)](#) -
[LogCollectedFees\(total.feeO,total.fee1\)](#) - [LogUncollectedFees\(underlying.feeO,underlying.fee1\)](#)

Reentrancy in [ArrakisV2.withdrawArrakisBalance\(\)](#): External calls: -
[tokenO.safeTransfer\(arrakisTreasury,amountO\)](#) - [token1.safeTransfer\(arrakisTreasury,amount1\)](#) Event emitted
after the call(s): - [LogWithdrawArrakisBalance\(amountO,amount1\)](#)

Reentrancy in [ArrakisV2.burn\(BurnLiquidity\[\],uint256,address\)](#): External calls: -
[tokenO.safeTransfer\(receiver_,amountO\)](#) - [token1.safeTransfer\(receiver_,amount1\)](#) Event emitted after the call(s):
- [LogBurn\(receiver_,burnAmount_,amountO,amount1\)](#)

Medium/Medium/divide-before-multiply

[ArrakisV2Factory._uint2str\(uint256\)](#) performs a multiplication on the result of a division: -[temp = \(48 + uint8\(_i -](#)
[\(_i / 10\) * 10\)\)](#)

Medium/Medium/uninitialized-local

[ArrakisV2Factory._preDeploy\(InitializePayload,bool\).result](#) is a local variable never initialized

[ArrakisV2.rebalance\(Range\[\],Rebalance,Range\[\]\).exist_scope_1](#) is a local variable never initialized

[ArrakisV2.burn\(BurnLiquidity\[\],uint256,address\).underlying](#) is a local variable never initialized

[ArrakisV2Resolver.standardBurnParams\(uint256,IArrakisV2\).underlying](#) is a local variable never initialized

[Manager.getManagerFeeBPS\(IManager\).feeBPS](#) is a local variable never initialized

[ArrakisV2Resolver._requireWeightUnder100\(RangeWeight\[\]\).i](#) is a local variable never initialized

[ArrakisV2Resolver.standardRebalance\(RangeWeight\[\],IArrakisV2\).numberOfPosLiq](#) is a local variable never
initialized

[ArrakisV2Resolver.standardRebalance\(RangeWeight\[\],IArrakisV2\).j](#) is a local variable never initialized

Medium/Medium/unused-return

[ArrakisV2Storage.blacklistRouters\(address\[\]\)](#) ignores return value by [routers.remove\(routers\[i\]\)](#).

[ArrakisV2Factory._preDeploy\(InitializePayload,bool\)](#) ignores return value by [this.getTokenName\(tokenO,token1\)](#).

[ArrakisV2Storage._whitelistRouters\(address\[\]\)](#) ignores return value by [routers.add\(routers\[i\]\)](#).

[ArrakisV2Storage._addPools\(uint24\[\],address,address\)](#) ignores return value by [_pools.add\(pool\)](#).

[ArrakisV2Helper.ranges\(IArrakisV2\)](#) ignores return value by [vault._ranges\(index\)](#).

[ArrakisV2Factory.deployVault\(InitializePayload,bool\)](#) ignores return value by [_vaults.add\(vault\)](#).

[Manager.getManagerFeeBPS\(IManager\)](#) ignores return value by [manager._managerFeeBPS\(\)](#).

[ArrakisV2Storage.removePools\(address\[\]\)](#) ignores return value by [pools.remove\(pools\[i\]\)](#).

Optimization/High/external-function

[getProxyImplementation\(address\)](#) should be declared external: - [ArrakisV2FactoryStorage.getProxyImplementation\(address\)](#).

[getProxyAdmin\(address\)](#) should be declared external: - [ArrakisV2FactoryStorage.getProxyAdmin\(address\)](#).

[vaults\(\)](#) should be declared external: - [ArrakisV2Factory.vaults\(\)](#).

[getAmountsForLiquidity\(int24,int24,int24,uint128\)](#) should be declared external: - [ArrakisV2Resolver.getAmountsForLiquidity\(int24,int24,int24,uint128\)](#).

[requireNotActiveRange\(IUniswapV3Factory,address,address,address,Range\)](#) should be declared external: - [Position.requireNotActiveRange\(IUniswapV3Factory,address,address,address,Range\)](#).

[validateTickSpacing\(address,Range\)](#) should be declared external: - [Pool.validateTickSpacing\(address,Range\)](#).

[rangeExist\(Range\[\],Range\)](#) should be declared external: - [Position.rangeExist\(Range\[\],Range\)](#).

[getManagerFeeBPS\(IManager\)](#) should be declared external: - [Manager.getManagerFeeBPS\(IManager\)](#).

[totalUnderlyingWithFees\(UnderlyingPayload\)](#) should be declared external: - [Underlying.totalUnderlyingWithFees\(UnderlyingPayload\)](#).

[computeMintAmounts\(uint256,uint256,uint256,uint256,uint256\)](#) should be declared external: - [Underlying.computeMintAmounts\(uint256,uint256,uint256,uint256,uint256\)](#).



v2-core

Tests result

45 passing (56s)

Tests coverage

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts\	97.33	53.57	100	97.08	
ArrakisV2.sol	97.06	52.86	100	97.14	272,286,304,377
ArrakisV2Beacon.sol	100	100	100	100	
ArrakisV2Factory.sol	96.97	50	100	97.06	113
ArrakisV2Helper.sol	100	100	100	100	
ArrakisV2Resolver.sol	96.43	58.33	100	95.08	105,106,247
contracts\abstract\	91.3	59.09	90	91.18	
ArrakisV2FactoryStorage.sol	100	50	100	100	
ArrakisV2Storage.sol	88.68	60.53	84.62	88.46	... 185,187,189
contracts\functions\	100	50	100	100	
FArrakisV2Factory.sol	100	50	100	100	
contracts\libraries\	94.12	62.5	100	94.12	
Manager.sol	66.67	100	100	66.67	12
Pool.sol	100	100	100	100	
Position.sol	100	50	100	100	
Underlying.sol	94.64	66.67	100	94.64	233,307,317

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts\structs\	100	100	100	100	
SArrakisV2.sol	100	100	100	100	
SArrakisV2Helper.sol	100	100	100	100	
All files	95.78	56.08	97.01	95.65	

v2-palm

Tests result

83 passing (58s)

Tests coverage

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
contracts\	100	60	100	100	
PALMManager.sol	100	50	100	100	
PALMTerms.sol	100	61.11	100	100	
contracts\abstracts\	87.88	82.43	86.79	89.93	
PALMManagerStorage.sol	91.03	86.36	87.5	91.57	... 253,254,326
PALMTermsStorage.sol	83.33	76.67	85.71	87.5	... 229,231,255
contracts\functions\	81.82	30	85.71	81.82	
FPALMTerms.sol	81.82	30	85.71	81.82	44,48
contracts\interfaces\	100	100	100	100	
IArrakisV2.sol	100	100	100	100	
IArrakisV2Beacon.sol	100	100	100	100	
IArrakisV2Factory.sol	100	100	100	100	
IArrakisV2Resolver.sol	100	100	100	100	

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
IManager.sol	100	100	100	100	
IPALMManager.sol	100	100	100	100	
IPALMTerms.sol	100	100	100	100	
contracts\structs\	100	100	100	100	
SPALMManager.sol	100	100	100	100	
SPALMTerms.sol	100	100	100	100	
All files	91.44	73.08	82.72	92.58	

STATE
MIND