



Arrakis Finance v2 PALM Audit Report

Oct 28, 2022





Table of Contents

Summary	2
Overview	3
Issues	4
[WP-M1] Unsafe <code>ERC20.approve()</code>	4
[WP-M2] Unsafe <code>ERC20.transfer()</code>	6
[WP-L3] Unbounded loop may cause <code>_addVault()</code> to malfunction due to out-of-gas	7
[WP-M4] <code>PALMManager._preExec()</code> Lack of internal accounting for accumulated rebalancing fee	8
[WP-S5] An alternative implementation of <code>renewTerm()</code>	10
[WP-S6] A simpler and condition controlled implementation of <code>decreaseLiquidity()</code>	13
[WP-S7] An alternative implementation of <code>increaseLiquidity()</code>	18
[WP-I8] Multiple methods on <code>PALMTerms</code> will result in a rug-like behavior	22
[WP-L9] Wrong function name	26
[WP-N10] <code>PALMTermsStorage#noLeftOver</code> Misleading modifier Name	27
[WP-N11] Unnecessary type casting	28
[WP-G12] Using <code>EnumerableSet.AddressSet</code> can avoid unnecessary storage reads	31
[WP-N13] Unused imports	33
Appendix	34
Disclaimer	35



Summary

This report has been prepared for Arrakis Finance v2 PALM Audit Report smart contract, to discover issues and vulnerabilities in the source code of their Smart Contract as well as any contract dependencies that were not part of an officially recognized library. A comprehensive examination has been performed, utilizing Static Analysis and Manual Review techniques.

The auditing process pays special attention to the following considerations:

- Testing the smart contracts against both common and uncommon attack vectors.
- Assessing the codebase to ensure compliance with current best practices and industry standards.
- Ensuring contract logic meets the specifications and intentions of the client.
- Cross referencing contract structure and implementation against similar smart contracts produced by industry leaders.
- Thorough line-by-line manual review of the entire codebase by industry experts.



Overview

Project Summary

Project Name	Arrakis Finance v2 PALM Audit Report
Codebase	https://github.com/ArrakisFinance/v2-palm
Commit	fe25734501e6cc7f429d34bbf8f70d86845c51c9
Language	Solidity

Audit Summary

Delivery Date	Oct 28, 2022
Audit Methodology	Static Analysis, Manual Review
Total Issues	13



[WP-M1] Unsafe `ERC20.approve()`

Medium

Issue Description

[https://github.com/ArrakisFinance/v2-palm/blob/](https://github.com/ArrakisFinance/v2-palm/blob/fe25734501e6cc7f429d34bbf8f70d86845c51c9/contracts/PALMTerms.sol#L40-L113)

[fe25734501e6cc7f429d34bbf8f70d86845c51c9/contracts/PALMTerms.sol#L40-L113](https://github.com/ArrakisFinance/v2-palm/blob/fe25734501e6cc7f429d34bbf8f70d86845c51c9/contracts/PALMTerms.sol#L40-L113)

```
40  function openTerm(SetupPayload calldata setup_, uint256 mintAmount_)
41      external
42      payable
43      override
44      noLeftOver(setup_.token0, setup_.token1)
45      returns (address vault)
46  {
    @@ 47,103 @@
104
105      setup_.token0.approve(vault, setup_.amount0);
106      setup_.token1.approve(vault, setup_.amount1);
107
108      vaultV2.setRestrictedMint(address(this));
109
110      vaultV2.mint(mintAmount_, address(this));
111
112      emit SetupVault(setup_.owner, vault);
113  }
```

`PALMTerms#openTerm()` calls `IERC20.approve()` before the call to `vaultV2.mint()`.

However, there are many Weird ERC20 Tokens that won't work correctly using the standard `IERC20` interface.

For example, the `USDT` token on mainnet does not return a bool on the `approve()` method.

As a result, when calling `IERC20(USDT).approve()`, the transaction will revert with an error: "function returned an unexpected amount of data", as the expected return data is a bool, but it actually does not return any data.



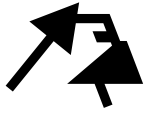
This means that `PALMTerms#openTerm()` doesn't work well with one of the most popular tokens.
`increaseLiquidity()` has the same issue.

Recommendation

Calling `safeApprove(0)` first and then `safeApprove(amount)` .

Status

✓ Fixed



[WP-M2] Unsafe `ERC20.transfer()`

Medium

Issue Description

Calling `ERC20.transfer()` directly without using `SafeERC20.transfer()` will result in reverts for some of the most popular tokens, e.g. USDT, BNB, OMG.

See: Weird ERC20 Tokens

https:

[//github.com/ArrakisFinance/v2-palm/blob/fe25734501e6cc7f429d34bbf8f70d86845c51c9/contracts/abstracts/PALMTermsStorage.sol#L45-L53](https://github.com/ArrakisFinance/v2-palm/blob/fe25734501e6cc7f429d34bbf8f70d86845c51c9/contracts/abstracts/PALMTermsStorage.sol#L45-L53)

```
45     modifier noLeftOver(IERC20 token0_, IERC20 token1_) {
46         uint256 token0Balance = token0_.balanceOf(address(this));
47         uint256 token1Balance = token1_.balanceOf(address(this));
48         _;
49         uint256 leftOver0 = token0_.balanceOf(address(this)) - token0Balance;
50         uint256 leftOver1 = token1_.balanceOf(address(this)) - token1Balance;
51         if (leftOver0 > 0) token0_.transfer(msg.sender, leftOver0);
52         if (leftOver1 > 0) token1_.transfer(msg.sender, leftOver1);
53     }
```

Recommendation

Consider using OpenZeppelin's `SafeERC20` library.

Status

✓ Fixed



[WP-L3] Unbounded loop may cause `_addVault()` to malfunction due to out-of-gas

Low

Issue Description

https:

[//github.com/ArrakisFinance/v2-palm/blob/fe25734501e6cc7f429d34bbf8f70d86845c51c9/contracts/abstracts/PALMTermsStorage.sol#L249-L258](https://github.com/ArrakisFinance/v2-palm/blob/fe25734501e6cc7f429d34bbf8f70d86845c51c9/contracts/abstracts/PALMTermsStorage.sol#L249-L258)

```
249 function _addVault(address creator_, address vault_) internal {
250     address[] storage vaultsOfCreator = vaults[creator_];
251
252     for (uint256 i = 0; i < vaultsOfCreator.length; i++) {
253         require(vaultsOfCreator[i] != vault_, "PALMTerms: vault exist");
254     }
255
256     vaultsOfCreator.push(vault_);
257     emit AddVault(creator_, vault_);
258 }
```

An attacker can add a lot of vaults under the victim's address (`creator_`).

Once the length of `vaultsOfCreator` is large enough, and the gas cost of function `_addVault()` can exceed the block limit, making it impossible to `addVault()` .

Recommendation

See [Recommendation] of [WP-G12].

Status

✓ Fixed



[WP-M4] `PALMManager._preExec()` Lack of internal accounting for accumulated rebalancing fee

Medium

Issue Description

<https://github.com/ArrakisFinance/v2-palm/blob/fe25734501e6cc7f429d34bbf8f70d86845c51c9/contracts/PALMManager.sol#L49-L64>

```
49     function _preExec(address vault_, uint256 feeAmount_)
50         internal
51         returns (uint256 balance)
52     {
53         VaultInfo memory vaultInfo = vaults[vault_];
54         require(
55             vaultInfo.balance >= feeAmount_,
56             "PALMManager: Not enough balance to pay fee"
57         );
58         balance = vaultInfo.balance - feeAmount_;
59
60         // update lastRebalance time
61         // solhint-disable-next-line not-rely-on-time
62         vaults[vault_].lastRebalance = block.timestamp;
63         vaults[vault_].balance = balance;
64     }
```

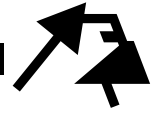
In the current implementation, a `feeAmount_` parameter will be passed by Operators when calling `rebalance()`, the `feeAmount_` will be charged from the vault's balance.

However, there is no internal accounting to record the total accumulated fee amount for the whole system or one particular operator.

There is no method to withdraw the fees either.

Recommendation

Consider adding a storage variable called `operatorBalances` to record the operators' fee earnings:

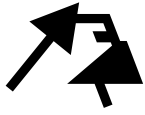


```
49 function _preExec(address vault_, uint256 feeAmount_)
50     internal
51     returns (uint256 balance)
52 {
53     VaultInfo memory vaultInfo = vaults[vault_];
54     require(
55         vaultInfo.balance >= feeAmount_,
56         "PALMManager: Not enough balance to pay fee"
57     );
58     balance = vaultInfo.balance - feeAmount_;
59
60     operatorBalances[msg.sender] += feeAmount_;
61
62     // update lastRebalance time
63     // solhint-disable-next-line not-rely-on-time
64     vaults[vault_].lastRebalance = block.timestamp;
65     vaults[vault_].balance = balance;
66 }
```

And also add a new method for the operators to withdraw their earnings.

Status

✓ Fixed



[WP-S5] An alternative implementation of `renewTerm()`

Issue Description

The current implementation of `renewTerm()` will pull all the funds back and then reinit the vault, which is not necessary.

[https://github.com/ArrakisFinance/v2-palm/blob/](https://github.com/ArrakisFinance/v2-palm/blob/fe25734501e6cc7f429d34bbf8f70d86845c51c9/contracts/PALMTerms.sol#L180-L220)

[fe25734501e6cc7f429d34bbf8f70d86845c51c9/contracts/PALMTerms.sol#L180-L220](https://github.com/ArrakisFinance/v2-palm/blob/fe25734501e6cc7f429d34bbf8f70d86845c51c9/contracts/PALMTerms.sol#L180-L220)

```
180  function renewTerm(IArrakisV2 vault_)
181      external
182      override
183      noLeftOver(vault_.token0(), vault_.token1())
184  {
185      IPALMManager manager_ = IPALMManager(manager);
186      require( // solhint-disable-next-line not-rely-on-time
187          manager_.getVaultInfo(address(vault_)).termEnd < block.timestamp,
188          "PALMTerms: term not ended."
189      );
190      IPALMManager(manager).renewTerm(address(vault_));
191
192      (uint256 amount0, uint256 amount1, uint256 balance) = _burn(
193          vault_,
194          address(this),
195          resolver
196      );
197
198      uint256 emolumentAmt0 = _getEmolument(amount0, emolument);
199      uint256 emolumentAmt1 = _getEmolument(amount1, emolument);
200      vault_.token0().approve(address(vault_), amount0 - emolumentAmt0);
201      vault_.token1().approve(address(vault_), amount1 - emolumentAmt1);
202      if (emolumentAmt0 > 0)
203          vault_.token0().safeTransfer(termTreasury, emolumentAmt0);
204      if (emolumentAmt1 > 0)
205          vault_.token1().safeTransfer(termTreasury, emolumentAmt1);
206      {
207          Inits memory inits;
208          (inits.init0, inits.init1) = _getInits(
209              balance,
210              amount0 - emolumentAmt0,
211              amount1 - emolumentAmt1
```

```

212         );
213
214         vault_.setInits(inits.init0, inits.init1);
215     }
216
217     vault_.mint(balance, address(this));
218
219     emit RenewTerm(address(vault_), emolumentAmt0, emolumentAmt1);
220 }

```

Recommendation

```

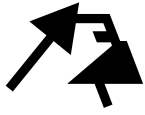
180 function renewTerm(IArrakisV2 vault_)
181     external
182     override
183     noLeftOver(vault_.token0(), vault_.token1())
184 {
185     IPALMManager manager_ = IPALMManager(manager);
186     require( // solhint-disable-next-line not-rely-on-time
187         manager_.getVaultInfo(address(vault_)).termEnd < block.timestamp,
188         "PALMTerms: term not ended."
189     );
190     IPALMManager(manager).renewTerm(address(vault_));
191
192     uint256 balance = IERC20(address(vault_)).balanceOf(this);
193     uint256 emolumentShares = _getEmolument(balance, emolument);
194
195     BurnLiquidity[] memory burnPayload = resolver.standardBurnParams(
196         emolumentShares,
197         vault_
198     );
199     (uint256 emolumentAmt0, uint256 emolumentAmt1) = vault_.burn(burnPayload,
200         emolumentShares, termTreasury);
201
202     emit RenewTerm(address(vault_), emolumentAmt0, emolumentAmt1);
203 }

```



Status

✓ Fixed

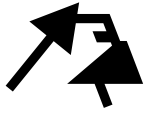


[WP-S6] A simpler and condition controlled implementation of `decreaseLiquidity()`

Issue Description

<https://github.com/ArrakisFinance/v2-palm/blob/fe25734501e6cc7f429d34bbf8f70d86845c51c9/contracts/PALMTerms.sol#L205-L287>

```
205  function decreaseLiquidity(
206      DecreaseBalance calldata decreaseBalance_,
207      uint256 mintAmount_
208  )
209      external
210      override
211      noLeftOver(
212          decreaseBalance_.vault.token0(),
213          decreaseBalance_.vault.token1()
214      )
215  {
216      _requireMintNotZero(mintAmount_);
217      _requireIsOwner(vaults[msg.sender], address(decreaseBalance_.vault));
218
219      address me = address(this);
220
221      (uint256 amount0, uint256 amount1, ) = _burn(
222          decreaseBalance_.vault,
223          me,
224          resolver
225      );
226      require(
227          decreaseBalance_.amount0 < amount0,
228          "PALMTerms: send back amount0 > amount0"
229      );
230      require(
231          decreaseBalance_.amount1 < amount1,
232          "PALMTerms: send back amount1 > amount1"
233      );
234
235      uint256 emolumentAmt0 = _getEmolument(
236          decreaseBalance_.amount0,
237          emolument
```



```
238     );
239     uint256 emolumentAmt1 = _getEmolument(
240         decreaseBalance_.amount1,
241         emolument
242     );
243
244     {
245         IERC20 token0 = decreaseBalance_.vault.token0();
246         IERC20 token1 = decreaseBalance_.vault.token1();
247
248         if (emolumentAmt0 > 0)
249             token0.safeTransfer(termTreasury, emolumentAmt0);
250         if (emolumentAmt1 > 0)
251             token1.safeTransfer(termTreasury, emolumentAmt1);
252
253         token0.safeTransfer(
254             decreaseBalance_.to,
255             decreaseBalance_.amount0 - emolumentAmt0
256         );
257         token1.safeTransfer(
258             decreaseBalance_.to,
259             decreaseBalance_.amount1 - emolumentAmt1
260         );
261         token0.approve(
262             address(decreaseBalance_.vault),
263             amount0 - decreaseBalance_.amount0
264         );
265         token1.approve(
266             address(decreaseBalance_.vault),
267             amount1 - decreaseBalance_.amount1
268         );
269     }
270     {
271         (uint256 init0, uint256 init1) = _getInits(
272             mintAmount_,
273             amount0 - decreaseBalance_.amount0,
274             amount1 - decreaseBalance_.amount1
275         );
276         decreaseBalance_.vault.setInits(init0, init1);
277     }
278
279     decreaseBalance_.vault.mint(mintAmount_, me);
280
```



```
281     emit DecreaseLiquidity(  
282         msg.sender,  
283         address(decreaseBalance_.vault),  
284         emolumentAmt0,  
285         emolumentAmt1  
286     );  
287 }
```

The current implementation will almost always success, even if the price has been changed and therefore changed the ratio/amounts of tokens pulled out with the transaction.

That's because it's pulling all the funds back first and taking the desired amounts, unless the caller is pulling all or at least a majority of the liquidity, this will most certainly succeed.

This can be a problem if the caller has a very specific intention with this `decreaseLiquidity()` call:

PoC

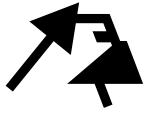
Given:

- The fair market price of ETH is 2000 USDC;
1. A huge swap has skewed the price in the Uni v3 pool from 2000 USDC to 1000 USDC;
 2. The owner of the PALM identified that as an arb opportunity and called `decreaseLiquidity()` to pull more ETH out and take a profit;
 3. Other arb txs have been minted before the `decreaseLiquidity()` tx;
 4. The owner of the PALM ends up with an undesirable price for the removal of liquidity.

Recommendation

We propose a new design that requires the caller to specify the amount of shares to burn and minimum amounts as slippage control:

```
205     function decreaseLiquidity(  
206         RemoveLiquidityData memory decreaseBalance_ // from  
207         `./structs/SArrakisV2Router.sol`  
208     )  
209     external
```

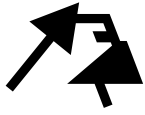



```
209     override
210     {
211         _requireIsOwner(vaults[msg.sender], address(decreaseBalance_.vault));
212
213         BurnLiquidity[] memory burnPayload = resolver.standardBurnParams(
214             decreaseBalance_.burnAmount,
215             decreaseBalance_.vault
216         );
217
218         (uint256 amount0, uint256 amount1) = decreaseBalance_.vault.burn(
219             burnPayload,
220             decreaseBalance_.burnAmount,
221             address(this)
222         );
223
224         require(
225             amount0 >= decreaseBalance_.amount0Min &&
226             amount1 >= decreaseBalance_.amount1Min,
227             "PALMTerms: received below minimum"
228         );
229
230         uint256 emolumentAmt0;
231         uint256 emolumentAmt1;
232
233         if (amount0 > 0) {
234             IERC20 token0 = decreaseBalance_.vault.token0();
235
236             emolumentAmt0 = _getEmolument(
237                 amount0,
238                 emolument
239             );
240             token0.safeTransfer(termTreasury, emolumentAmt0);
241             token0.safeTransfer(
242                 decreaseBalance_.receiver,
243                 amount0 - emolumentAmt0
244             );
245         }
246
247         if (amount1 > 0) {
248             IERC20 token1 = decreaseBalance_.vault.token1();
249
250             emolumentAmt1 = _getEmolument(
251                 amount1,
```

```
252         emolument
253     );
254     token1.safeTransfer(termTreasury, emolumentAmt1);
255     token1.safeTransfer(
256         decreaseBalance_.receiver,
257         amount1 - emolumentAmt1
258     );
259 }
260
261 emit DecreaseLiquidity(
262     msg.sender,
263     address(decreaseBalance_.vault),
264     emolumentAmt0,
265     emolumentAmt1
266 );
267 }
```

Status

✓ Fixed



[WP-S7] An alternative implementation of `increaseLiquidity()`

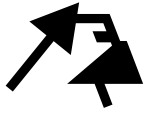
Issue Description

Considering the fact that the vault is 100% owned by the vault's owner, `increaseLiquidity()` can be simplified to plain token transfers to the vault.

[https://github.com/ArrakisFinance/v2-palm/blob/](https://github.com/ArrakisFinance/v2-palm/blob/fe25734501e6cc7f429d34bbf8f70d86845c51c9/contracts/PALMTerms.sol#L116-L177)

[fe25734501e6cc7f429d34bbf8f70d86845c51c9/contracts/PALMTerms.sol#L116-L177](https://github.com/ArrakisFinance/v2-palm/blob/fe25734501e6cc7f429d34bbf8f70d86845c51c9/contracts/PALMTerms.sol#L116-L177)

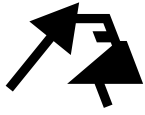
```
116  function increaseLiquidity(  
117      IncreaseBalance calldata increaseBalance_, // memory instead of calldata to  
118      set values  
119      uint256 mintAmount_  
120  )  
121  external  
122  override  
123  noLeftOver(  
124      increaseBalance_.vault.token0(),  
125      increaseBalance_.vault.token1()  
126  )  
127  {  
128      _requireMintNotZero(mintAmount_);  
129      _requireProjectAllocationGtZero(  
130          increaseBalance_.projectTknIsTknZero,  
131          increaseBalance_.amount0,  
132          increaseBalance_.amount1  
133      );  
134      _requireIsOwner(vaults[msg.sender], address(increaseBalance_.vault));  
135      (uint256 amount0, uint256 amount1, ) = _burn(  
136          increaseBalance_.vault,  
137          address(this),  
138          resolver  
139      );  
140  
141      // Transfer to termTreasury the project token emolment.  
142      increaseBalance_.vault.token0().safeTransferFrom(  
143          msg.sender,  
144          address(this),  
145          increaseBalance_.amount0  
146      );
```



```
147     increaseBalance_.vault.token1().safeTransferFrom(
148         msg.sender,
149         address(this),
150         increaseBalance_.amount1
151     );
152
153     increaseBalance_.vault.token0().approve(
154         address(increaseBalance_.vault),
155         increaseBalance_.amount0 + amount0
156     );
157
158     increaseBalance_.vault.token1().approve(
159         address(increaseBalance_.vault),
160         increaseBalance_.amount1 + amount1
161     );
162
163     {
164         Inits memory inits;
165         (inits.init0, inits.init1) = _getInits(
166             mintAmount_,
167             increaseBalance_.amount0 + amount0,
168             increaseBalance_.amount1 + amount1
169         );
170
171         increaseBalance_.vault.setInits(inits.init0, inits.init1);
172     }
173
174     increaseBalance_.vault.mint(mintAmount_, address(this));
175
176     emit IncreaseLiquidity(msg.sender, address(increaseBalance_.vault));
177 }
```

<https://github.com/ArrakisFinance/vault-v2-core/blob/702a343cea74dc9a551d6098926448c8e62d798b/contracts/ArrakisV2.sol#L52-L98>

```
52     function mint(uint256 mintAmount_, address receiver_)
53         external
54         nonReentrant
55         returns (uint256 amount0, uint256 amount1)
56     {
```




```
@@ 57,81 @@  
82  
83     amount0 = FullMath.mulDivRoundingUp(mintAmount_, current0, denominator);  
84     amount1 = FullMath.mulDivRoundingUp(mintAmount_, current1, denominator);  
85  
86     _mint(receiver_, mintAmount_);  
87  
88     // transfer amounts owed to contract  
89     if (amount0 > 0) {  
90         token0.safeTransferFrom(msg.sender, me, amount0);  
91     }  
92     if (amount1 > 0) {  
93         token1.safeTransferFrom(msg.sender, me, amount1);  
94     }  
95  
96     emit LogUncollectedFees(fee0, fee1);  
97     emit LogMint(receiver_, mintAmount_, amount0, amount1);  
98 }
```

Recommendation

<https://github.com/ArrakisFinance/v2-palm/blob/fe25734501e6cc7f429d34bbf8f70d86845c51c9/contracts/PALMTerms.sol#L116-L177>

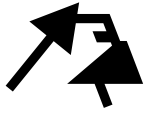
```
116 function increaseLiquidity(  
117     IncreaseBalance calldata increaseBalance_ // memory instead of calldata to set  
118     )  
119     external  
120     override  
121     {  
122         increaseBalance_.vault.token0().safeTransferFrom(  
123             msg.sender,  
124             increaseBalance_.vault,  
125             increaseBalance_.amount0  
126         );  
127         increaseBalance_.vault.token1().safeTransferFrom(  
128             msg.sender,  
129             increaseBalance_.vault,  
130             increaseBalance_.amount1
```



```
131     );  
132  
133     emit IncreaseLiquidity(msg.sender, address(increaseBalance_.vault));  
134 }
```

Status

✓ Fixed



[WP-I8] Multiple methods on `PALMTerms` will result in a rug-like behavior

Informational

Issue Description

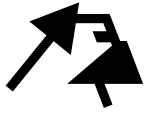
There are 3 methods on `PALMTerms` that are intended for regular operations, which should not result in a significant change in liquidity:

- `increaseLiquidity()`
- `renewTerm()`
- `decreaseLiquidity()`

The current implementation of these functions will actually withdraw ALL the liquidity first, then call `vault.mint()` to put the money back:

<https://github.com/ArrakisFinance/v2-palm/blob/fe25734501e6cc7f429d34bbf8f70d86845c51c9/contracts/PALMTerms.sol#L223-L305>

```
223  function decreaseLiquidity(  
224      DecreaseBalance calldata decreaseBalance_,  
225      uint256 mintAmount_  
226  )  
227      external  
228      override  
229      noLeftOver(  
230          decreaseBalance_.vault.token0(),  
231          decreaseBalance_.vault.token1()  
232      )  
233  {  
234      _requireMintNotZero(mintAmount_);  
235      _requireIsOwner(vaults[msg.sender], address(decreaseBalance_.vault));  
236  
237      address me = address(this);  
238  
239      (uint256 amount0, uint256 amount1, ) = _burn(  
240          decreaseBalance_.vault,  
241          me,  
242          resolver
```

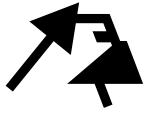


```
243     );  
    @@ 244,295 @@  
296  
297     decreaseBalance_.vault.mint(mintAmount_, me);  
298  
299     emit DecreaseLiquidity(  
300         msg.sender,  
301         address(decreaseBalance_.vault),  
302         emolumentAmt0,  
303         emolumentAmt1  
304     );  
305 }
```

However, `vault.mint()` won't actually put the liquidity back to the Uniswap v3 position, until the next `rebalance()` :

<https://github.com/ArrakisFinance/vault-v2-core/blob/702a343cea74dc9a551d6098926448c8e62d798b/contracts/ArrakisV2.sol#L52-L98>

```
52  function mint(uint256 mintAmount_, address receiver_)  
53      external  
54      nonReentrant  
55      returns (uint256 amount0, uint256 amount1)  
56  {  
57      require(mintAmount_ > 0, "MA");  
58      require(  
59          restrictedMint == address(0) || msg.sender == restrictedMint,  
60          "R"  
61      );  
62      address me = address(this);  
63      uint256 totalSupply = totalSupply();  
64      (  
65          uint256 current0,  
66          uint256 current1,  
67          uint256 fee0,  
68          uint256 fee1  
69      ) = totalSupply > 0  
70          ? UnderlyingHelper.totalUnderlyingWithFees(  
71              UnderlyingPayload({  
72                  ranges: ranges,
```

```
73         factory: factory,
74         token0: address(token0),
75         token1: address(token1),
76         self: me
77     })
78 )
79 : (init0, init1, 0, 0);
80 uint256 denominator = totalSupply > 0 ? totalSupply : 1 ether;
81 /// @dev current0 and current1 include fees and left over (but not admin
balances)
82
83     amount0 = FullMath.mulDivRoundingUp(mintAmount_, current0, denominator);
84     amount1 = FullMath.mulDivRoundingUp(mintAmount_, current1, denominator);
85
86     _mint(receiver_, mintAmount_);
87
88     // transfer amounts owed to contract
89     if (amount0 > 0) {
90         token0.safeTransferFrom(msg.sender, me, amount0);
91     }
92     if (amount1 > 0) {
93         token1.safeTransferFrom(msg.sender, me, amount1);
94     }
95
96     emit LogUncollectedFees(fee0, fee1);
97     emit LogMint(receiver_, mintAmount_, amount0, amount1);
98 }
```

This creates a gap period between the `increaseLiquidity()` / `renewTerm()` / `decreaseLiquidity()` transaction and the `rebalance()` .

During that period, it would look just like the protocol pulled the rug, and the liquidity will be much thinner than before (depends on the percentage owned by the protocol).

We believe this is undesirable and also unnecessary.

Recommendation

See [WP-S5], [WP-S6], [WP-S7].



Status

✓ Fixed



[WP-L9] Wrong function name

Low

Issue Description

<https://github.com/ArrakisFinance/vault-v2-agreement/blob/0b8fc8f69de3a1c662e623e5faa0b86603a5bc41/contracts/functions/FTerms.sol#L95-L97>

```
95  function _requireAddressNotZero(uint256 mintAmount_) pure {  
96      require(mintAmount_ > 0, "Terms: mintAmount zero.");  
97  }
```

Should be named `_requireMintAmountNotZero` .

Status

✓ Fixed



[WP-N10] PALMTermsStorage#noLeftOver Misleading modifier

Name

Issue Description

https:

[//github.com/ArrakisFinance/v2-palm//blob/fe25734501e6cc7f429d34bbf8f70d86845c51c9/contracts/abstracts/PALMTermsStorage.sol#L45-L53](https://github.com/ArrakisFinance/v2-palm/blob/fe25734501e6cc7f429d34bbf8f70d86845c51c9/contracts/abstracts/PALMTermsStorage.sol#L45-L53)

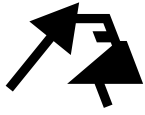
```
45     modifier noLeftOver(IERC20 token0_, IERC20 token1_) {
46         uint256 token0Balance = token0_.balanceOf(address(this));
47         uint256 token1Balance = token1_.balanceOf(address(this));
48         _;
49         uint256 leftOver0 = token0_.balanceOf(address(this)) - token0Balance;
50         uint256 leftOver1 = token1_.balanceOf(address(this)) - token1Balance;
51         if (leftOver0 > 0) token0_.transfer(msg.sender, leftOver0);
52         if (leftOver1 > 0) token1_.transfer(msg.sender, leftOver1);
53     }
```

`noLeftOver` can be misunderstood as `require(leftOverX == 0, "...")` .

Consider renaming to `collectLeftOver` or `sweepLeftOver` .

Status

✓ Fixed



[WP-N11] Unnecessary type casting

Issue Description

https:

[//github.com/ArrakisFinance/v2-palm/blob/fe25734501e6cc7f429d34bbf8f70d86845c51c9/contracts/abstracts/PALMTermsStorage.sol#L231-L243](https://github.com/ArrakisFinance/v2-palm/blob/fe25734501e6cc7f429d34bbf8f70d86845c51c9/contracts/abstracts/PALMTermsStorage.sol#L231-L243)

```
231 function withdrawVaultBalance(  
232     address vault_,  
233     uint256 amount_,  
234     address payable to_  
235 ) external override requireAddressNotZero(vault_) {  
236     address vaultAddr = address(vault_);  
237     IPALMManager manager_ = IPALMManager(manager);  
238     (uint256 balance, , , , ) = manager_.vaults(vaultAddr);  
239     _requireIsOwner(vaults[msg.sender], vaultAddr);  
240     manager_.withdrawVaultBalance(vault_, amount_, to_);  
241  
242     emit LogWithdrawVaultBalance(msg.sender, vaultAddr, to_, balance);  
243 }
```

https:

[//github.com/ArrakisFinance/v2-palm/blob/fe25734501e6cc7f429d34bbf8f70d86845c51c9/contracts/abstracts/PALMTermsStorage.sol#L179-L199](https://github.com/ArrakisFinance/v2-palm/blob/fe25734501e6cc7f429d34bbf8f70d86845c51c9/contracts/abstracts/PALMTermsStorage.sol#L179-L199)

```
179 function setVaultData(address vault_, bytes calldata data_)  
180     external  
181     override  
182     requireAddressNotZero(vault_)  
183 {  
184     address vaultAddr = address(vault_);  
185     _requireIsOwnerOrDelegate(  
186         delegateByVaults[vault_],  
187         vaults[msg.sender],  
188         vaultAddr  
189     );  
190     IPALMManager(manager).setVaultData(vault_, data_);  
191  
192     emit LogSetVaultData(  
193         vault_,  
194         data_,  
195         vaultAddr,  
196         msg.sender,  
197         vaults[msg.sender].balance,
```



```
193         delegateByVaults[vault_] != address(0)
194         ? delegateByVaults[vault_]
195         : msg.sender,
196         vaultAddr,
197         data_
198     );
199 }
```


https:

[//github.com/ArrakisFinance/v2-palm/blob/fe25734501e6cc7f429d34bbf8f70d86845c51c9/contracts/abstracts/PALMTermsStorage.sol#L201-L221](https://github.com/ArrakisFinance/v2-palm/blob/fe25734501e6cc7f429d34bbf8f70d86845c51c9/contracts/abstracts/PALMTermsStorage.sol#L201-L221)

```
201 function setVaultStratByName(address vault_, string calldata strat_)
202     external
203     override
204     requireAddressNotZero(vault_)
205 {
206     address vaultAddr = address(vault_);
207     _requireIsOwnerOrDelegate(
208         delegateByVaults[vault_],
209         vaults[msg.sender],
210         vaultAddr
211     );
212     IPALMManager(manager).setVaultStraByName(vault_, strat_);
213
214     emit LogSetVaultStratByName(
215         delegateByVaults[vault_] != address(0)
216         ? delegateByVaults[vault_]
217         : msg.sender,
218         vaultAddr,
219         strat_
220     );
221 }
```

https:

[//github.com/ArrakisFinance/v2-palm/blob/fe25734501e6cc7f429d34bbf8f70d86845c51c9/contracts/abstracts/PALMTermsStorage.sol#L223-L229](https://github.com/ArrakisFinance/v2-palm/blob/fe25734501e6cc7f429d34bbf8f70d86845c51c9/contracts/abstracts/PALMTermsStorage.sol#L223-L229)



```
223 function setDelegate(address vault_, address delegate_) external override {  
224     address vaultAddr = address(vault_);  
225     _requireIsOwner(vaults[msg.sender], vaultAddr);  
226     _setDelegate(vault_, delegate_);  
227  
228     emit LogSetDelegate(msg.sender, vaultAddr, delegate_);  
229 }
```

Status

✓ Fixed



[WP-G12] Using `EnumerableSet.AddressSet` can avoid unnecessary storage reads

Gas

Issue Description

https:

[//github.com/ArrakisFinance/v2-palm/blob/fe25734501e6cc7f429d34bbf8f70d86845c51c9/contracts/abstracts/PALMTermsStorage.sol#L249-L258](https://github.com/ArrakisFinance/v2-palm/blob/fe25734501e6cc7f429d34bbf8f70d86845c51c9/contracts/abstracts/PALMTermsStorage.sol#L249-L258)

```
249     function _addVault(address creator_, address vault_) internal {
250         address[] storage vaultsOfCreator = vaults[creator_];
251
252         for (uint256 i = 0; i < vaultsOfCreator.length; i++) {
253             require(vaultsOfCreator[i] != vault_, "PALMTerms: vault exist");
254         }
255
256         vaultsOfCreator.push(vault_);
257         emit AddVault(creator_, vault_);
258     }
```

<https://github.com/ArrakisFinance/v2-palm/blob/fe25734501e6cc7f429d34bbf8f70d86845c51c9/contracts/abstracts/PALMTermsStorage.sol#L36>

```
36     mapping(address => address[]) public vaults;
```

Recommendation

```
249     function _addVault(address creator_, address vault_) internal {
250         EnumerableSet.AddressSet storage vaultsOfCreator = vaults[creator_];
251
252         require(!vaultsOfCreator.contains(vault_), "PALMTerms: vault exist");
253
254         vaultsOfCreator.add(vault_);
255         emit AddVault(creator_, vault_);
```




256 }

Status

✓ Fixed



[WP-N13] Unused imports

Issue Description

contracts/PALMManager.sol

- "IPALMManager" is unused
- "IUniswapV3Pool" is unused
- "SafeERC20" is unused
- "IERC20" is unused

contracts/PALMTerms.sol

- "_requireTokenMatch" is unused
- "_requireIsOwnerOrDelegate" is unused

contracts/abstracts/PALMTermsStorage.sol

- "FullMath" is unused
- "SafeCast" is unused
- "_getInits" is unused
- "_requireTokenMatch" is unused
- "_getEmolument" is unused
- "_requireProjectAllocationGtZero" is unused
- "_requireTknOrder" is unused
- "_burn" is unused

Status

✓ Fixed



Appendix

Timeliness of content

The content contained in the report is current as of the date appearing on the report and is subject to change without notice, unless indicated otherwise by WatchPug; however, WatchPug does not guarantee or warrant the accuracy, timeliness, or completeness of any report you access using the internet or other means, and assumes no obligation to update any information following publication.



Disclaimer

This report is based on the scope of materials and documentation provided for a limited review at the time provided. Results may not be complete nor inclusive of all vulnerabilities. The review and this report are provided on an as-is, where-is, and as-available basis. You agree that your access and/or use, including but not limited to any associated services, products, protocols, platforms, content, and materials, will be at your sole risk. Smart Contract technology remains under development and is subject to unknown risks and flaws. The review does not extend to the compiler layer, or any other areas beyond the programming language, or other programming aspects that could present security risks. A report does not indicate the endorsement of any particular project or team, nor guarantee its security. No third party should rely on the reports in any way, including for the purpose of making any decisions to buy or sell a product, service or any other asset. To the fullest extent permitted by law, we disclaim all warranties, expressed or implied, in connection with this report, its content, and the related services and products and your use thereof, including, without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement. We do not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party through the product, any open source or third-party software, code, libraries, materials, or information linked to, called by, referenced by or accessible through the report, its content, and the related services and products, any hyperlinked websites, any websites or mobile applications appearing on any advertising, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third-party providers of products or services. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate. FOR AVOIDANCE OF DOUBT, THE REPORT, ITS CONTENT, ACCESS, AND/OR USAGE THEREOF, INCLUDING ANY ASSOCIATED SERVICES OR MATERIALS, SHALL NOT BE CONSIDERED OR RELIED UPON AS ANY FORM OF FINANCIAL, INVESTMENT, TAX, LEGAL, REGULATORY, OR OTHER ADVICE.