



PALADIN
BLOCKCHAIN SECURITY

Smart Contract Security Assessment

Final Report

For Trader Joe (v2 Vaults)

11 May 2023



paladinsec.co



info@paladinsec.co

Table of Contents

Table of Contents	2
Disclaimer	4
1 Overview	5
1.1 Summary	5
1.2 Contracts Assessed	6
1.3 Findings Summary	7
1.3.1 General Issues	8
1.3.2 VaultFactory	8
1.3.3 BaseVault	9
1.3.4 SimpleVault	9
1.3.5 OracleVault	10
1.3.6 Strategy	10
1.3.7 CloneExtension	10
1.3.8 Distribution	11
1.3.9 Math	11
1.3.10 Revision Audit	11
2 Findings	12
2.1 General Issues	12
2.1.1 Issues & Recommendations	13
2.2 VaultFactory	14
2.2.1 Privileged Functions	17
2.2.2 Issues & Recommendations	18
2.3 BaseVault	21
2.3.1 Privileged Functions	22
2.3.2 Issues & Recommendations	23
2.4 SimpleVault	33
2.4.1 Privileged Functions	33
2.4.2 Issues & Recommendations	34

2.5 OracleVault	35
2.5.1 Privileged Functions	35
2.5.2 Issues & Recommendations	36
2.6 Strategy	38
2.6.1 Privileged Functions	39
2.6.2 Issues & Recommendations	40
2.7 CloneExtension	49
2.7.1 Issues & Recommendations	49
2.8 Distribution	50
2.8.1 Issues & Recommendations	51
2.9 Math	52
2.9.1 Issues & Recommendations	52
2.10 Revision Audit	53
2.10.1 Issues & Recommendations	55



Disclaimer

Paladin Blockchain Security ("Paladin") has conducted an independent audit to verify the integrity of and highlight any vulnerabilities or errors, intentional or unintentional, that may be present in the codes that were provided for the scope of this audit. This audit report does not constitute agreement, acceptance or advocacy for the Project that was audited, and users relying on this audit report should not consider this as having any merit for financial advice in any shape, form or nature. The contracts audited do not account for any economic developments that may be pursued by the Project in question, and that the veracity of the findings thus presented in this report relate solely to the proficiency, competence, aptitude and discretion of our independent auditors, who make no guarantees nor assurance that the contracts are completely free of exploits, bugs, vulnerabilities or deprecation of technologies. Further, this audit report shall not be disclosed nor transmitted to any persons or parties on any objective, goal or justification without due written assent, acquiescence or approval by Paladin.

All information provided in this report does not constitute financial or investment advice, nor should it be used to signal that any persons reading this report should invest their funds without sufficient individual due diligence regardless of the findings presented in this report. Information is provided 'as is', and Paladin is under no covenant to the completeness, accuracy or solidity of the contracts audited. In no event will Paladin or its partners, employees, agents or parties related to the provision of this audit report be liable to any parties for, or lack thereof, decisions and/or actions with regards to the information provided in this audit report.

Cryptocurrencies and any technologies by extension directly or indirectly related to cryptocurrencies are highly volatile and speculative by nature. All reasonable due diligence and safeguards may yet be insufficient, and users should exercise considerable caution when participating in any shape or form in this nascent industry.

The audit report has made all reasonable attempts to provide clear and articulate recommendations to the Project team with respect to the rectification, amendment and/or revision of any highlighted issues, vulnerabilities or exploits within the contracts provided. It is the sole responsibility of the Project team to sufficiently test and perform checks, ensuring that the contracts are functioning as intended, specifically that the functions therein contained within said contracts have the desired intended effects, functionalities and outcomes of the Project team.

Paladin retains the right to re-use any and all knowledge and expertise gained during the audit process, including, but not limited to, vulnerabilities, bugs, or new attack vectors. Paladin is therefore allowed and expected to use this knowledge in subsequent audits and to inform any third party, who may or may not be our past or current clients, whose projects have similar vulnerabilities. Paladin is furthermore allowed to claim bug bounties from third-parties while doing so.

1 Overview

This report has been prepared for Trader Joe's v2 Vault contracts on the Avalanche and Arbitrum networks. Paladin provides a user-centred examination of the smart contracts to look for vulnerabilities, logic errors or other issues from both an internal and external perspective.

1.1 Summary

Project Name	Trader Joe
URL	http://traderjoexyz.com/
Platform	Avalanche & Arbitrum
Language	Solidity
Preliminary contracts	https://github.com/traderjoe-xyz/joe-v2-vault/commit/a67c592b9be135ee7191d61d46b1411e91ea8055
Resolution #1	https://github.com/traderjoe-xyz/joe-v2-vault/tree/b675c5d07bbd3acb53b34d482f810580061d1686
Resolution #2	https://github.com/traderjoe-xyz/joe-v2-vault/tree/95c919b6f16299dc0f8322d74202fba801a08f38/src

1.2 Contracts Assessed

Name	Contract	Live Code Match
VaultFactory	0x613aDdd55D30BF626c3009e9DAC66222346Ae760	✓ MATCH
BaseVault	Dependency	✓ MATCH
SimpleVault	0xc8240C9e7eA8f94aC12da06b1B17be3e0F415a6D	✓ MATCH
OracleVault	0xFA92D54bAD5Bb85f59e73B557Ea9BBBD7b3B7F266	✓ MATCH
Strategy	0xAF705A05789BEeCB5d4983453da7D35f4585d92c	✓ MATCH
CloneExtension	Not used – using Clone contract from JoeV2	UNUSED
Distribution	Not used	UNUSED
Math	Library	✓ MATCH



1.3 Findings Summary

Severity	Found	Resolved	Partially Resolved	Acknowledged (no change made)
● High	1	1	-	-
● Medium	0	-	-	-
● Low	19	14	-	5
● Informational	17	11	-	6
Total	37	26	-	11

Classification of Issues

Severity	Description
● High	Exploits, vulnerabilities or errors that will certainly or probabilistically lead towards loss of funds, control, or impairment of the contract and its functions. Issues under this classification are recommended to be fixed with utmost urgency.
● Medium	Bugs or issues with that may be subject to exploit, though their impact is somewhat limited. Issues under this classification are recommended to be fixed as soon as possible.
● Low	Effects are minimal in isolation and do not pose a significant danger to the project or its users. Issues under this classification are recommended to be fixed nonetheless.
● Informational	Consistency, syntax or style best practices. Generally pose a negligible level of risk, if any.

1.3.1 General Issues

ID	Severity	Summary	Status
01	INFO	General governance privileges	✓ RESOLVED

1.3.2 VaultFactory

ID	Severity	Summary	Status
02	LOW	Incompatibility with DEX V2.1	✓ RESOLVED
03	LOW	_getName can return an incorrect name in future deployments	✓ RESOLVED
04	LOW	feeRecipient can be address(0)	✓ RESOLVED
05	INFO	Lack of validation	✓ RESOLVED
06	INFO	Typographical errors	✓ RESOLVED
07	INFO	Hardcoded dataFeeds	ACKNOWLEDGED



1.3.3 BaseVault

ID	Severity	Summary	Status
08	HIGH	Pending queues are burnable by anyone	✓ RESOLVED
09	LOW	Incompatibility with tokens that have a fee on transfer	ACKNOWLEDGED
10	LOW	cancelQueuedWithdrawal sends the shares to the wrong address	✓ RESOLVED
11	LOW	redeemQueuedWithdrawal and redeemQueuedWithdrawalNative allow anyone to redeem any queued withdrawal	✓ RESOLVED
12	LOW	Possible revert in executeQueuedWithdrawals due to safe downcasting	✓ RESOLVED
13	LOW	Possible revert in multiplication overflow in emergencyWithdraw	✓ RESOLVED
14	LOW	Lack of balance check after token transfer in recoverERC20 can allow draining of underlying tokens with double entry points	✓ RESOLVED
15	LOW	Pending fee logic is unused in DEX V2.1	✓ RESOLVED
16	LOW	Whitelisted users not publicly visible	ACKNOWLEDGED
17	INFO	Unnecessary duplicate condition check in recoverERC20	✓ RESOLVED
18	INFO	Typographical errors	✓ RESOLVED
19	INFO	Frontend phishing risk	ACKNOWLEDGED
20	INFO	Lack of events for recoverERC20	✓ RESOLVED

1.3.4 SimpleVault

ID	Severity	Summary	Status
21	LOW	First depositor determines the ratio	ACKNOWLEDGED

1.3.5 OracleVault

ID	Severity	Summary	Status
22	LOW	Lack of staleness check for Chainlink price	✓ RESOLVED
23	INFO	Lack of verification that both oracle feeds have the same decimals	✓ RESOLVED

1.3.6 Strategy

ID	Severity	Summary	Status
24	LOW	Possible revert in multiplication overflow in _withdraw	✓ RESOLVED
25	LOW	Multiple spots are incompatible with DEX V2.1	✓ RESOLVED
26	LOW	Fee for queued amounts is rounded down	✓ RESOLVED
27	LOW	Attacker can DoS the rebalance	ACKNOWLEDGED
28	LOW	Lack of use of safeApprove in swap	✓ RESOLVED
29	LOW	rebalance and swap can have very high slippage	ACKNOWLEDGED
30	INFO	AUM fee is charged on entire deposits, not only profits	ACKNOWLEDGED
31	INFO	Lack of non-zero validation	ACKNOWLEDGED
32	INFO	Typographical issues	✓ RESOLVED
33	INFO	Lack of events for several functions	ACKNOWLEDGED

1.3.7 CloneExtension

No issues found.

1.3.8 Distribution

ID	Severity	Summary	Status
34	INFO	Typographical errors	✓ RESOLVED

1.3.9 Math

No issues found.

1.3.10 Revision Audit

ID	Severity	Summary	Status
35	INFO	Composition fee might be applied	
36	INFO	Typographical errors	
37	INFO	Unnecessary casting to uint24	

2 Findings

2.1 General Issues

The issues in this section apply to the protocol as a whole.



2.1.1 Issues & Recommendations

Issue #01 General governance privileges	
Severity	 INFORMATIONAL
Description	<p>Privileged addresses for these contract(s) have multiple governance privileges that might result in a loss of user funds. These privileges are necessary in order to provide flexibility with regards to rebalancing and other methodologies. Since the TraderJoe team is widely known and reputable in the space, we do not expect any malicious behavior.</p> <p>Moreover, the team has a proven track record in keeping user funds safe — the TraderJoe BMCJ Farms and the whole lending protocol were running for months under upgradeable proxies without any issues.</p>
Recommendation	Consider being strict with the selection of the privileged addresses, especially the <code>_defaultOperator</code> in the factory should be a trusted party, ideally a multi-signature wallet.
Resolution	 RESOLVED <code>_defaultOperator</code> and owner will be granted to the Trader Joe multi-signature wallet.

2.2 VaultFactory

`VaultFactory` is used for the deployment of two different vault types and one strategy type.

The difference between the two vault types, `OracleVault` and `SimpleVault`, is the methodology of the share calculation. While `SimpleVault` executes a raw calculation based on the underlying token amounts, `OracleVault` calculates the shares denominated in the value of `tokenY`.

There is only one type of strategy available, the `Default` type. `VaultFactory` allows for either a complete deployment of the strategy and the vault via `createSimpleVaultAndStrategy` and `CreateOracleVaultAndStrategy`. With these two functions, a vault and a corresponding strategy gets deployed and are immediately linked to each other.

Another method is to simply deploy either an oracle vault or a simple vault via `createSimpleVault` and `createOracleVault`, then deploy a strategy via `createDefaultStrategy` and link the vault and the corresponding strategy via `linkVaultToStrategy`.

It is important to mention that every deployment is done via the `ClonesWithImmutableArgs` implementation which allows the deployment of clones from a preset implementation with immutable arguments which are then padded to the calldata of each `delegatecall`.

For the simple vault, the immutable arguments are the following:

1. `lbPair`
2. `tokenX`
3. `tokenY`
4. `decimalsX`
5. `decimalsY`

For the oracle vault, the following arguments are used:

1. `lbPair`
2. `tokenX`
3. `tokenY`
4. `decimalsX`
5. `decimalsY`
6. `dataFeedX`
7. `dataFeedY`

Whereas for the strategy, the following arguments are used:

1. `vault`
2. `lbPair`
3. `tokenX`
4. `tokenY`

This logic allows the vault's strategy to be changed, whereas the strategy always has an immutable vault address assigned.

Besides deployment, the factory also has other functions such as changing the implementation for the `oracleVault`, `simpleVault` and `strategy`, which gives the owner the freedom to modify future deployments.

Various privileged calls can be made to the strategy and the vault such as changing the strategy operator, setting and resetting the `PendingAumAnnualFee`, changing the `feeRecipient`, pausing and resuming deposits, setting the vault into an emergency mode and handling whitelist functionality.

All in all, the factory gives the owner various governance privileges which should be carefully used.



2.2.1 Privileged Functions

- `setVaultImplementation`
- `setStrategyImplementation`
- `setDefaultOperator`
- `setOperator`
- `setPendingAumAnnualFee`
- `resetPendingAumAnnualFee`
- `setFeeRecipient`
- `createOracleVaultAndDefaultStrategy`
- `createSimpleVaultAndDefaultStrategy`
- `createSimpleVault`
- `createOracleVault`
- `createDefaultStrategy`
- `linkVaultToStrategy`
- `setWhitelistState`
- `addToWhitelist`
- `removeFromWhitelist`
- `pauseDeposits`
- `resumeDeposits`
- `setEmergencyMode`
- `recoverERC20`
- `transferOwnership`
- `renounceOwnership`



2.2.2 Issues & Recommendations

Issue #02	Incompatibility with DEX V2.1
Severity	 LOW SEVERITY
Description	<p>Since these vaults should solely run with DEX V2.1, all interactions should implement the correct function selectors:</p> <p><u>Lines 234/235/256/257/271/272/290/291/303/304</u></p> <p>The correct function selector is getTokenX, getTokenY.</p> <p><u>Line 482</u></p> <p>The correct function selector is getBinStep.</p>
Recommendation	Consider using the correct function selectors and correct parameters in order to be compatible with DEX V2.1.
Resolution	 RESOLVED The functions were updated to work with v2.1

Issue #03	_getName can return an incorrect name in future deployments
Severity	 LOW SEVERITY
Description	Since there might be the possibility that the TraderJoe team implements other vault types in the future, there might be an issue with the _getName function because this function only checks if the type is a simple vault and if not, it will simply return "Oracle".
Recommendation	Consider explicitly checking if the type is VaultType.Oracle and only then should it return "Oracle".
Resolution	 RESOLVED

Issue #04	feeRecipient can be address(0)
Severity	 LOW SEVERITY
Description	setFeeRecipient allows the owner to set the feeRecipient to any address. However, if it is accidentally set to address(0), it will block the rebalance function in the strategy contract, which essentially means that no deposits can be executed.
Recommendation	Consider validating the parameter accordingly.
Resolution	 RESOLVED

Issue #05	Lack of validation
Severity	 INFORMATIONAL
Description	<p><u>Line 55</u> <code>_wnative = wnative;</code></p> <p>The wnative address should be validated against address(0).</p> <p><u>Line 62</u> <code>function initialize(address owner)</code></p> <p>The owner address should be validated against address(0).</p> <p><u>Line 181</u> <code>function setDefaultOperator</code></p> <p>defaultOperator should be validated against address(0).</p>
Recommendation	Consider fixing the above issues.
Resolution	 RESOLVED <p>There is a check that wnative does not revert on balanceOf call, and that owner is not address(0). defaultOperator to can still be set to address(0) in case it is necessary.</p>

Issue #06		Typographical errors
Severity	 INFORMATIONAL	
Description	<p><u>Line 25</u></p> <p>* - SimpleVault: This vault is used to interact with pairs where one of the token has no oracle. Deposits needs to be</p> <p>This should be "Deposits need to be" and "one of the tokens"</p> <p><u>Line 27</u></p> <p>* - OracleVault: This vault is used to interact with pairs where both tokens have an oracle. Deposits doesn't need to</p> <p>This should be "Deposits don't need to".</p>	
Recommendation	Consider fixing the above issues.	
Resolution	 RESOLVED	

Issue #07		Hardcoded dataFeeds
Severity	 INFORMATIONAL	
Description	<p>For the Oracle vault deployment, both dataFeeds are hard coded using the ClonesWithImmutableArgs pattern. This makes the dataFeeds unchangeable in case of unexpected events. Since we already acknowledged that this protocol is under full governance control, it might make sense to implement logic which allows the dataFeeds to be changed.</p>	
Recommendation	Consider if this is potentially valuable, if yes, consider implementing logic that initializes the dataFeeds and makes them changeable instead of hardcoding it into the proxy contract.	
Resolution	 ACKNOWLEDGED	

2.3 BaseVault

BaseVault represents the base of the vault architecture.

Users can deposit tokens in order to get shares minted, and the logic of the share calculation is separately handled in the SimpleVault and the OracleVault.

BaseVault is quite different compared to traditional vaults: users can deposit either tokenX and tokenY or just one of both tokens, and BaseVault also contains functionality to deposit AVAX directly, if one of both tokens is WAVAX.

The `_previewShares` function within the SimpleVault or the OracleVault will then take care of the share and amount calculation.

Once a user has successfully deposited and received the desired amount of shares, the tokens will then sit in the Strategy contract which contains additional functionality to add them to a LP pair in order to accumulate fees.

Unlike most vaults, users cannot withdraw whenever they want to. The withdrawal process involves multiple steps and starts with the `queueWithdrawal` function, which allows users to request a withdrawal. They will simply transfer their shares in and receive an allocation for the current withdrawal round. After that is done, the function `executeQueuedWithdrawals` needs to be called from the Strategy contract which then allocates the correct amount of tokenX and tokenY to the current round based on the total queued withdrawals. Users can then claim their allocated tokens via `redeemQueuedWithdrawal` which simply calculates the user's pro-rated amount based on the allocated amount, user shares and `totalShares`.

If a user decides to keep the tokens within the vault, one can simply call `cancelQueuedWithdrawal` which then simply transfers back the shares and removes the allocation for the current round. It is important that this is only possible for the current round, that means if `executeQueuedWithdrawals` has

been called, it is no longer possible to call `cancelQueuedWithdrawal` because `executeQueuedWithdrawals` initiates a new round.

`BaseVault` also implements a whitelist mode where only specific users can deposit. The owner of the Factory can switch this mode on and off and handle the whitelisted addresses.

2.3.1 Privileged Functions

- `setStrategy`
- `pauseDeposits`
- `resumeDeposits`
- `setWhitelistState`
- `addToWhitelist`
- `removeFromWhitelist`
- `setEmergencyMode`
- `recoverERC20`



2.3.2 Issues & Recommendations

Issue #08	Pending queues are burnable by anyone
Severity	 HIGH SEVERITY
Description	<p>Currently the withdraw flow is as follows:</p> <ol style="list-style-type: none">1. User requests a withdrawal via <code>queueWithdraw</code>2. Admin rebalances the strategy which then calls <code>executeQueuedWithdrawals</code> which assigns and transfers the token amounts in, corresponding to the queued shares3. User executes withdrawal via <code>redeemQueuedWithdrawal</code> and receives the desired tokens. <p>However, there is a dangerous flaw involved into this flow: before the admin rebalances the strategy, <code>redeemQueuedWithdrawal</code> can be called, however, since there is no amount yet assigned for this round, the <code>queuedWithdrawals</code> of the user will be set to zero but no tokens are transferred to the user. This will effectively result in a loss of user funds and locks the tokens in the contract. This issue is amplified more because <code>redeemQueuedWithdrawal</code> allows redemptions for other addresses, effectively opening the opportunity for a malicious attacker to burn all queued shares.</p> <p>The same issue applies for the native function.</p>
Recommendation	<p>Consider adding a check that the current round cannot be redeemed.</p> <p>Additionally, consider adding a check to revert if both <code>amountX</code> and <code>amountY</code> are both 0, since there should never be a valid scenario where they are both 0.</p>
Resolution	 RESOLVED
	<p>A check to revert if the round to redeem is the current round was added. The safety check to revert if both amounts are zero was added.</p>

Issue #09**Incompatibility with tokens that have a fee on transfer****Severity** LOW SEVERITY**Description**

If either token of the LB pair has a fee on transfer, it would result in loss of tokens deposited due to fees.

During deposits, the underlying tokens would be transferred and incur fees twice, once from the user to the strategy and another from the strategy to LB pair contract.

During rebalances, the underlying tokens would be transferred and incur fees twice too, once when withdrawing from the LB pair contract to the strategy, and then again when depositing back from the strategy to the LB pair.

Finally, for queued withdrawal, the underlying tokens would be transferred and incur fees twice, once when sending the queued amounts from the strategy to the vault, and another when the user redeems the shares for the underlying token.

Recommendation

Consider not supporting fee on transfer tokens by checking the difference between the token balance of X and Y tokens in the strategy before and after the `safeTransferFrom` call and ensure that it is the same as the amount of tokens to be transferred.

Resolution ACKNOWLEDGED

The team has indicated that the vaults will not support tokens with a fee on transfer. No change has been made.

Issue #10	cancelQueuedWithdrawal sends the shares to the wrong address
Severity	 LOW SEVERITY
Description	cancelQueuedWithdrawal expects the shares to be sent to the address which was given as an input parameter of recipient. However, it is sent back to the msg.sender instead.
Recommendation	Consider transferring the shares as desired to the recipient or simply remove the recipient parameter and send it back to msg.sender.
Resolution	 RESOLVED recipient was removed.

Issue #11	redeemQueuedWithdrawal and redeemQueuedWithdrawalNative allow anyone to redeem any queued withdrawal
Severity	 LOW SEVERITY
Description	<p>redeemQueuedWithdrawal and redeemQueuedWithdrawalNative both take in the recipient argument, which allows for the vault contract to send out the underlying tokens to the recipient address.</p> <p>This can be an issue for users who are in countries where the timing on when you receive a token affects taxation. A user who may have a queued withdrawal ready for redeeming, but could be waiting for the next taxation year to withdraw, would be affected if someone calls the function to redeem on behalf of the user before that.</p>
Recommendation	Consider using msg.sender for redemption of queued withdrawals.
Resolution	 RESOLVED The redeem functions are now restricted to the user itself or the factory as the factory only batch redeems for the user.

Issue #12**Possible revert in executeQueuedWithdrawals due to safe downcasting****Severity** LOW SEVERITY**Description**

Within executeQueuedWithdrawals, the token balances of token X and Y are used to downcast from uint256 to uint128, which reverts if it exceeds the maximum value of uint128. This is possible as ERC20 token balances are of the uint256 datatype.

L632-638

// Get the amount of tokens received by the vault after executing the withdrawals.

```
uint256 receivedX = _tokenX().balanceOf(address(this)) -  
totalAmountX;
```

```
uint256 receivedY = _tokenY().balanceOf(address(this)) -  
totalAmountY;
```

// Update the total amounts of tokens in the vault.

```
_totalAmountX = (totalAmountX + receivedX).safe128();
```

```
_totalAmountY = (totalAmountY + receivedY).safe128();
```

While this would require a huge number of tokens to exceed the uint128 maximum value, it could be possible for underlying tokens that are either a large supply, or have a high number of decimals.

Recommendation

Consider using uint256 for _totalAmountX and _totalAmountY.

Resolution RESOLVED

uint256 is now used.

Issue #13**Possible revert in multiplication overflow in emergencyWithdraw****Severity** LOW SEVERITY**Description**

In emergencyWithdraw, the amountX and Y are calculated via the following:

L587-592

```
(uint256 balanceX, uint256 balanceY) =  
_getBalances(IStrategy(address(0)));  
uint256 totalShares = totalSupply();
```

```
uint256 amountX = balanceX * shares / totalShares;  
uint256 amountY = balanceY * shares / totalShares;
```

balanceX and balanceY are dependent on the balance in the vault contract. If there is a case where the result of balanceX or balanceY multiplied by shares exceeds the maximum value uint256, a revert would occur.

Recommendation

Use mulDivRoundDown as done with other sections of the code such as _previewAmounts.

Resolution RESOLVED

Issue #14**Lack of balance check after token transfer in recoverERC20 can allow draining of underlying tokens with double entry points****Severity** LOW SEVERITY**Description**

In the case of a double entry point such as TUSD, since the alternative entry point would have a contract address that does not match `_tokenX()` or `_tokenY()`, it would bypass the balance check even though the alternative entry point contract address used would still allow withdrawing of the same underlying token.

L767 - L777

```
// Checks that the amount of token X to be recovered is not  
from any withdrawal. This will simply revert  
if the vault is in emergency mode.  
if (token == _tokenX() && (strategy == address(0) ||  
token.balanceOf(address(this)) < _totalAmountX + amount)) {  
    revert BaseVault__InvalidToken();  
}
```

```
// Checks that the amount of token Y to be recovered is not  
from any withdrawal. This will simply revert  
if the vault is in emergency mode.  
if (token == _tokenY() && (strategy == address(0) ||  
token.balanceOf(address(this)) < _totalAmountY + amount)) {  
    revert BaseVault__InvalidToken();  
}
```

Reference: <https://blog.openzeppelin.com/compound-tusd-integration-issue-retrospective/>

Recommendation

Add an additional check after the token transfer is done to ensure there is sufficient X and Y token balances.

```
if (strategy == address(0)) {  
    if ( _tokenX().balanceOf(address(this)) < _totalAmountX  
|| _tokenY().balanceOf(address(this)) < _totalAmountY) {  
        revert BaseVault__InvalidToken();  
    }  
}
```

Resolution RESOLVED

Issue #15	Pending fee logic is unused in DEX V2.1
Severity	 LOW SEVERITY
Description	<p>The vault still assumes that the fees will be accumulated and then claimed. However, the DEX V2.1 autocompounds the fees, which means they will not be externally claimable. Therefore all logic related to claiming fees can be removed:</p> <p><u>Line 232</u></p> <pre>function getPendingFees() public</pre>
Recommendation	Consider removing the pending fee logic.
Resolution	 RESOLVED <p>The function has been updated to work with v2.1.</p>

Issue #16	Whitelisted users not publicly visible
Severity	 LOW SEVERITY
Description	There is no easy check to find all whitelisted users. This might be an important variable for users to inspect.
Recommendation	Consider adding all whitelisted users to an enumerableSet.
Resolution	 ACKNOWLEDGED



Severity

 INFORMATIONAL

Description

In recoverERC20, if the token's address is the vault address, the following check is done:

L779-791

```
if (token == this) {  
    uint256 excessStrategy = balanceOf(strategy) -  
    getCurrentTotalQueuedWithdrawal();  
  
    // If the token is the vault's token, the remaining  
    amount must be greater than the minimum shares.  
    if (token == this && balanceOf(address(this)) +  
    excessStrategy < amount + _SHARES_PRECISION) {  
        revert BaseVault__BurnMinShares();  
    }  
  
    // Allow to recover vault tokens that were mistakenly  
    sent to the strategy.  
    if (excessStrategy > 0) { //  
        _transfer(strategy, address(this), excessStrategy);  
    }  
}
```

The token == this comparison in the inner if block is unnecessary as it is already done one the out if block.

Recommendation

Remove the unnecessary comparison in the inner if block.

Resolution

 RESOLVED

Severity

 INFORMATIONAL

Description

Line 7

```
import {Initializable} from "openzeppelin-upgradeable/proxy/
utils/Initializable.sol";
```

This import is unused.

Line 10

```
import {ILBToken} from "joe-v2/interfaces/ILBToken.sol";
```

This import is unused .

Line 16

```
import {IAggregatorV3} from "../interfaces/
IAggregatorV3.sol";
```

It is not necessary to import the aggregatorV3 interface here, the OracleVault already imports it.

Line 109

```
* (3 addresses and 2 bytes of lenth), so this function
should never be called.
```

It should be lengths instead of lenth.

Line 870

```
* LB and the unclaiemd and redeemed fees.
```

This should be called accumulated fees because the fees are automatically added to the LP pair.

Recommendation

Consider fixing the above issues.

Resolution

 RESOLVED

Issue #19	Frontend phishing risk
Severity	 INFORMATIONAL
Description	<p>The contract allows several functions to be called with other addresses, most notably the <code>queueWithdrawal</code> function which assigns the shares of the <code>msg.sender</code> to the <code>input</code> address.</p> <p>The same issue potentially applies to the <code>cancelQueuedWithdrawal</code> function.</p> <p>This exposes a risk if the <code>TraderJoe</code> frontend is ever compromised or if phishing sites are present.</p>
Recommendation	Consider keeping an eye on such phishing sites and warn the community during such events.
Resolution	 ACKNOWLEDGED

Issue #20	Lack of events for <code>recoverERC20</code>
Severity	 INFORMATIONAL
Description	Functions that affect the status of sensitive variables should emit events as notifications.
Recommendation	Add events for the above function.
Resolution	 RESOLVED

2.4 SimpleVault

SimpleVault is a simple extension of the BaseVault which takes care of the share calculation. The logic was forked from AlphaShares and calculates the shares based on the desired input amounts and the underlying amounts of tokens, with the difference that the shares have a precision of $1e6$. It also ensures that the ratio is the same as the ratio of the underlying tokens.

2.4.1 Privileged Functions

- all privileged BaseVault functions



2.4.2 Issues & Recommendations

Issue #21	First depositor determines the ratio
Severity	 LOW SEVERITY
Description	The ratio will be based on the first depositor's amounts. This also means that if the first depositor decides to only add tokenX or tokenY, it will be impossible for others to add the corresponding token until liquidity has been added and swaps have been executed over the added bin in order to receive an underlying balance which includes both amounts.
Recommendation	The resolution is non-trivial. Consider if this is an issue which can be accepted, if not, the share calculation logic needs to be changed.
Resolution	 ACKNOWLEDGED

2.5 OracleVault

OracleVault is an extension of the BaseVault and handles the share calculation by calculating the value of tokenX and tokenY in tokenY and then calculates the shares proRata based on the underlying values.

2.5.1 Privileged Functions

- all privileged BaseVault functions



2.5.2 Issues & Recommendations

Issue #22	Lack of staleness check for Chainlink price
Severity	 LOW SEVERITY
Description	<p>In the case where the Chainlink price is stale and has not been updated for a period of time, using such a stale price can cause unexpected behavior.</p> <p>An example of this is the LUNA token which dropped below the Chainlink aggregator minimum price of \$0.01 (1000000), and Chainlink stopped updating the price. Protocols using the stale LUNA pricing ended up in monetary losses.</p>
Recommendation	<p>Define a buffer time which can be based on the Chainlink feed's heartbeat and some extra leeway, and check that the updatedAt time of the price has not exceeded the buffer.</p> <pre>if (block.timestamp - updatedAt > BUFFER) { revert StalePrice(); }</pre>
Resolution	 RESOLVED
	<p>A 24 hour buffer check since the last update time for a price feed has been added.</p>

Issue #23	Lack of verification that both oracle feeds have the same decimals
Severity	 INFORMATIONAL
Description	<p>For <code>_getPrice</code>, there is a following comment warning that the oracles have to both use the same decimals.</p> <p>* WARNING: Both oracles needs to return the same decimals and use the same quote currency.</p> <p>However, there is currently no check during deployment that enforces this check.</p> <p>While Chainlink feeds for token pricing in USD are generally using 8 decimals, there might be some feeds which use different decimals.</p>
Recommendation	At the point of deployment of the Oracle vault, it is recommended to add a sanity check that both are the same decimals.
Resolution	 RESOLVED
	This is now checked when setting the datafeeds in the factory.



2.6 Strategy

Strategy handles the interaction with the desired liquidity pair and takes care of adding liquidity, removing liquidity and rebalancing.

During the normal day-to-day operations, the flow follows a deterministic circle which starts with the `rebalance` function. The strategy operator and the default operator which is determined within the Factory can initiate a `rebalance` as often as they desire. This is the only entry point for adding liquidity which means that even for the first deposit, the whole rebalance flow must happen.

First, all LB tokens will be burned which results in removing the whole liquidity from all bins, and afterwards, the queued shares are fetched from the vault and the queued amount will then get calculated `proRata` from the queued shares, the total withdrawn amount and the total shares in existence.

The strategy then transfers the queued amounts to the vault and calls `executeQueuedWithdrawals` which assigns the correct amount for the current round and allows all users with queued shares to withdraw their desired amount. Then, the range will be set which consists of `newLower` and `newUpper`. These two variables are the inclusive bounds for the bin selection. In the next step, the contract calculates `distributionX` and `distributionY` based on the desired liquidity for each step using the `Distribution` library. Then in the final step, liquidity will be added based on the previously calculated `distributionX` and `distributionY` variables within the desired range, assuming the contract holds sufficient tokens for each position, which means if the contract only holds `tokenX`, it is obviously not possible to add any positions to the `activeBin` in form of `tokenY`, let alone adding any positions below the current `activeId`.

Therefore, the range, distribution and desired liquidity must be carefully validated before adding liquidity.

Within the rebalance function, an AuM fee applies to the whole liquidity. The assets-under-management fee was implemented by TraderJoe based the identically-named fee used in traditional finance systems. The fee can be set up to 25% per year, which equals 0.06849% per day. This fee applies during each rebalance but can only be taken for 1 day at a maximum, which means that if a rebalance only happens each week, the fee will be 0.06849% per week.

The `withdrawAll` function can be called by the vault in emergency mode, which is a special feature that was implemented to react to unpredictable events. This function simply withdraws the full liquidity, takes care of the queued logic and transfers the remaining amount back to the vault where users can withdraw their funds via `emergencyWithdraw`.

Additionally, a swap function was implemented which allows users to swap tokenX for tokenY or vice-versa using 1inch.

2.6.1 Privileged Functions

- `setOperator (factory)`
- `setPendingAumAnnualFee (factory)`
- `resetPendingAumAnnualFee (factory)`
- `withdrawAll (vault)`
- `rebalance (operator)`
- `swap (operator)`

2.6.2 Issues & Recommendations

Issue #24	Possible revert in multiplication overflow in <code>_withdraw</code>
Severity	 LOW SEVERITY
Description	<p>Within <code>_withdraw</code>, <code>amountX</code> and <code>amountY</code> are calculated by the following:</p> <pre><u>L774-776</u> (queuedAmountX, queuedAmountY) = totalShares == 0 queuedShares == 0 ? (0, 0) : (queuedShares * totalX / totalShares, queuedShares * totalY / totalShares)</pre> <p>If there is a case where the result of <code>totalX</code> or <code>totalY</code> multiplied by <code>queuedShares</code> exceeds the max value <code>uint256</code>, a revert would occur.</p>
Recommendation	Consider using <code>mulDivRoundDown</code> as done in other sections of the codebase.
Resolution	 RESOLVED

Description

The vault still assumes that the fees will be accumulated and then claimed. However, the DEX V2.1 autocompounds the fees, which means they will not be externally claimable. In V2.1, the pair contract does not have an interface or function implemented for `pendingFees` and `collectFees`. Attempts to call these functions on the Pair address will result in a revert:

- `_collectFees`
- `_getPendingFees`
- `_pair().getReservesAndId()`
- `safeBatchTransferFrom()`

The following functions rely on calling these functions:

- `_withdrawFromLB`
- `collectFees`
- `_getBalances`
- `getPendingFees`

Some function parameters are incompatible as well:

—

```
ILBPair(pair).mint(_getIds(lower, upper), distributionX,  
distributionY, address(this));
```

The compatible function is:

```
mint(address to, bytes32[] calldata liquidityConfigs,  
address refundTo)
```

—

```
ILBPair(pair).burn(ids, amounts, address(this));
```

The compatible function is:

```
function burn(address from, address to, uint256[]  
calldata ids, uint256[] calldata amountsToBurn)
```

Recommendation Consider adjusting the contract in order to be compatible with DEX V2.1.

Resolution  RESOLVED

Issue #26 Fee for queued amounts is rounded down

Severity  LOW SEVERITY

Description The fee which is deducted from the queuedAmount is rounded down:
$$(\text{feeX} * \text{queuedAmountX} - 1) / \text{totalBalanceX};$$

While this does not have a huge impact, it might still make more sense to round it up in order to increase the amount which is deducted.

Recommendation Consider rounding up the fee.

Resolution  RESOLVED

Description

Due to the logic on how liquidity is added, a malicious user can DoS the addition of liquidity if the vault uses the liquidity capital efficiently, i.e. `desiredL` array is near the overall liquidity value in the vault.

Consider the following scenario:

1. The vault has 100 tokenY 20 tokenX as balance
2. The operator wants to add liquidity over a range, let's say from 8_000_000 to 8_000_005
3. The current `activeId` is 8_000_005 and the price is exactly 1 i.e. 340282366920938463374607431768211456
4. The `binStep` is 100 aka 1% price change per bin (we do know that the price for this `binStep` and `id` is not 1, but just for display purpose we use that, as the example works with every `id` and price the same way)
5. The operator wants now to add to the elements [0;4] each a desired liquidity of 20 and for element 5 the desired liquidity of 40, which means that for the current `activeId` 20 tokenY and 20 tokenX will be left
6. While this calculation now perfectly works, an attacker can now frontrun the rebalance function by swapping tokenX for tokenY with the goal to move the bin down in order to decrease the price for tokenX.
7. Let's say this example moves the `activeBin` from 8_000_005 to 8_000_004, which then also decreases the start of our range by 1, if the slippage is given.
8. Given the desired liquidity with [20,20,20,40] and the contract balance of 100 tokenX and 20 tokenY, the liquidity addition will now fail because for the current `activeId`, the value of tokenY is now 0.99 instead of 1. In order to have a sufficient balance to fulfill the desired liquidity in the `activeId`, it now needs 21 tokenX and 20 tokenY.

All in all, due to the decrease of tokenX value in the current activeBin, the contract balance now must be 100 tokenY and 21 tokenX to be able to fulfill the desired liquidity.

Recommendation Consider carefully validating the transaction result of the rebalance function and, if necessary resend the transaction with different parameters. If this issue becomes too annoying, it might make sense to not utilize a too high percentage of the total contract balance and keep the slippage low, but high enough to counteract a DoS with the slippage change.

Resolution

 ACKNOWLEDGED

This is a known issue and the team will resend the transaction after some time. If this issue becomes too frequent, the team may scale desiredL down according to the different balance.

Issue #28 **Lack of use of safeApprove in swap**

Severity

 LOW SEVERITY

Description

In the swap function, the `_ONE_INCH_ROUTER` is granted `srcToken` allowance by the approve function.

L326

```
desc.srcToken.approve(address(_ONE_INCH_ROUTER),
desc.amount);
_ONE_INCH_ROUTER.swap(executor, desc, "", data);
desc.srcToken.approve(address(_ONE_INCH_ROUTER), 0);
```

If the `srcToken` is a token that does not conform to ERC20 standards (e.g. USDT on ETH network which does not return a bool), the approve call would not be successful.

Recommendation Use `safeApprove` from the `SafeERC20Upgradeable` contract.

Resolution

 RESOLVED

Issue #29**rebalance and swap can have very high slippage****Severity** LOW SEVERITY**Description**

The rebalance function allows for very high slippage via `slippageActiveId`. While we totally consider slippage as necessary, it might be valuable to limit it to a reasonable value.

If the slippage is accidentally ever set too high, an exploiter can frontrun that by swapping to a very distant bin which then would result in the vault adding liquidity to that bin. This liquidity can then be abused by the exploiter to steal one side of the reserves.

There is also the similar issue in swap, which only checks the `minReturnAmount` is non 0, but can still be an high slippage value amount.

Recommendation

Consider limiting the `slippageActiveId` and `minReturnAmount` to a reasonable value.

Resolution ACKNOWLEDGED

`slippageActiveId` will be a fixed parameter in the vault configuration database. `minReturnAmount` cannot be set to a reasonable value as it depends on the tokens and the current price, and the use of Chainlink's API for the amounts which is trusted.



Issue #30 AUM fee is charged on entire deposits, not only profits	
Severity	INFORMATIONAL
Description	<p>During each rebalance, if the AUM annual fee percentage is not set to 0, the fee is charged on the entire <code>totalBalanceX</code> and <code>totalBalanceY</code>. Thus, if the fees earned by the vault is unable to outperform the AUM annual fee percentage, it can result in capital loss for users.</p> <p>Moreover, a fee on the first deposit is applied as well during the rebalance.</p>
Recommendation	Consider openly communicating this with the community.
Resolution	ACKNOWLEDGED

Issue #31 Lack of non-zero validation	
Severity	INFORMATIONAL
Description	<p><code>setOperator</code> lacks a non-zero check which can result in issues if the factory is at one point unable to set the operator to a regular address again.</p> <p>If it is in fact desired to set the strategy operator to <code>address(0)</code>, consider introducing a <code>renounceOperator</code> function.</p>
Recommendation	Consider validating the function accordingly.
Resolution	ACKNOWLEDGED

Severity

 INFORMATIONAL

Description

Line 10

```
import {ILBToken} from "joe-v2/interfaces/ILBToken.sol";
```

This import is done twice.

Line 697

```
if (annualFee > 0 && block.timestamp > lastRebalance) {
```

The `block.timestamp > lastRebalance` condition has already been checked above.

Line 819

```
// If the range is empty, get the amount of tokens in the pair.
```

This should be called in the strategy.

Line 494 and 502

```
delta = desiredActiveId - uint24(activeId)
```

The `activeId` is already `uint24`.

Recommendation

Consider fixing the typographical errors.

Resolution

 RESOLVED

Issue #33	Lack of events for several functions
Severity	<div><div></div> INFORMATIONAL</div>
Description	<p>Functions that affect the status of sensitive variables should emit events as notifications.</p> <ul style="list-style-type: none">- swap- rebalance- withdrawAll- collectFees
Recommendation	Add events for the above functions.
Resolution	<div><div></div> ACKNOWLEDGED</div>



2.7 CloneExtension

The CloneExtension library is a simple extension of the Clone contract from Clones with immutable args which simply extends the reading of immutable args by uint16.

2.7.1 Issues & Recommendations

No issues found.



2.8 Distribution

The `Distribution` library is used within the `Strategy` contract and is responsible for computing distributions for a given amount of tokens. If liquidity is only meant to be added above the current `activeId`, the function `computeDistributionX` will take care of it, and if otherwise, the function `computeDistributionY` will take care of the calculation. If liquidity should be added over the full range including the `activeId`, the ratios of `tokenX` and `tokenY` for the `activeId` will be calculated to match the ratio of the current `activeBin` in order to ensure that the deposit does not cost any `compositionFee`.

The distributions for `X` and `Y` will be calculated regularly using `computeDistributionX` and `distributionY`.



2.8.1 Issues & Recommendations

Issue #34	Typographical errors
Severity	 INFORMATIONAL
Location	<u>Line 134</u> totalAmont
Description	This should be be spelled totalAmount.
Recommendation	Consider fixing the typographical errors.
Resolution	 RESOLVED

2.9 Math

The Math library is used throughout the project and simply takes care of calculating the max and min between two uint256 variables.

2.9.1 Issues & Recommendations

No issues found.



2.10 Revision Audit

After the initial audit, the Trader Joe team decided to adjust the rebalance mechanism in order to allow the addition of liquidity at a different ratio than the ratio of the current pair.

For example, with a stable vault, it is desired to deposit almost the entire vault's assets and the previous method does not allow them to do this unless the vault has the same ratio as the pair's.

While we recognize the necessity of this change, it still requires additional validation. The changes made were very reasonable:

1. The `rebalance` function was modified in order to take amounts and distributions as input parameters instead of the desired liquidity. The distributions parameter is provided as bytes whereas the `distributionX/Y` for each bin are packed in bytes16 each, and all bytes16 blocks together may result in 1e18 in order to align with the `PRECISION` variable and efficiently consume the desired amounts. The code properly validates that the length of distributions matches the range of the desired liquidity to add.
2. A new internal function `getLiquidityConfigs` was introduced which converts the bin range and the distributions into `liquidityConfigs`:
 - 1) 0-23: ID
 - 2) 24-87: DISTX
 - 3) 88-151: DISTY
 - 4) 152-255: EMPTY

The goal of this additional audit is to carefully validate the logic and identify any potential issues resulting from it.

Since the governance privileges were already acknowledged we will not focus on any potential further governance issues resulting from this increased flexibility. It is to be ensured that no funds are lost during the process of adding liquidity.



2.10.1 Issues & Recommendations

Issue #35	Composition fee might be applied
Severity	<div><div></div>INFORMATIONAL</div>
Description	<p>A change was made to allow the addition of custom defined <code>liquidityConfigs</code> instead of the previous design where the distribution library was used to calculate the distribution used for <code>liquidityConfigs</code>.</p> <p>While this gives the team more flexibility to fully utilize the entire <code>tokenX</code> and <code>tokenY</code> balance in strategy, it can result in composition fees being charged when the <code>mint</code> function is called in the LB pair. Whenever the ratio from the current <code>activeBin</code> is different from the ratio being added to the current <code>activeBin</code>, an implicit swap is executed.</p> <p>However, the so-called composition fee is applied on this swap, which is similar to the fee for normal swaps.</p> <p>This can result in a decrease of funds during the rebalance function.</p>
Recommendation	<p>Consider keeping this in mind when adding liquidity. Initially we recommended checking the changes of the value before and after adding liquidity in terms of USD. However, this change would be quite invasive and since we assume that Trader Joe wants full flexibility, we do not necessarily expect this to be implemented.</p>
Resolution	<div><div></div>ACKNOWLEDGED</div>

Issue #36 Typographical errors	
Severity	 INFORMATIONAL
Description	<p><u>Line 269</u></p> <pre>// Get the distributions and the amounts to deposit, this will take into account the composition of the // LB pair to avoid the active fee.</pre> <p>In does not account for the composition fee anymore.</p> <p><u>Line 482</u></p> <pre>* distributions = abi.encodePacked(uint64(distribX0), uint64(distribY0), uint64(distribX1), uint64(distribY1), ...]</pre> <p>The proper notation would be <code>abi.encodePacked(uint64(distribX0), uint64(distribY0), uint64(distribX1), uint64(distribY1), ...)</code> - using regular brackets.</p>
Recommendation	Consider fixing the typographical errors.
Resolution	 RESOLVED

Issue #37 Unnecessary casting to uint24	
Severity	 INFORMATIONAL
Location	<p><u>L460</u></p> <pre>delta = uint24(activeId) - desiredActiveId;</pre>
Description	activeId is already declared as a uint24, so there is no need to cast it again to uint24.
Recommendation	Consider removing the unnecessary cast.
Resolution	 RESOLVED



PALADIN
BLOCKCHAIN SECURITY