Tribe Turbo Audit Findings

1. Slurp function ERC4626 withdraw parameters reversed

Proof of Concept: Lines 263 & 264 of TurboSafe.sol

```
// If we have unaccrued fees, withdraw them from the Vault and transfer them to the Master.
if (protocolFeeAmount != 0) vault.withdraw(protocolFeeAmount, address(this), address(master));
```

The `withdraw` function in ERC4626.sol has the input parameters ordered as `function withdraw(uint256 amount, address to, address from)`. This means the "to" and "from" addresses are the reverse of what they should be in

Risk Breakdown

High risk, because the amount withdrawn will be sent in the wrong direction. This will likely cause a revert. If deployed in its current state to mainnet, a redeployment would be necessary.

Recommendation:

Swapping withdraw parameters to the following:

```
// If we have unaccrued fees, withdraw them from the Vault and transfer them to the Master.
if (protocolFeeAmount != 0) vault.withdraw(protocolFeeAmount, address(master), address(this));
```

2. nonReentrant modifier on internal function causes revert

The two TurboSafe.sol internal functions `beforeWithdraw` and `afterDeposit` override ERC4626 functions but add the nonReentrant modifier. When these internal nonReentrant functions are called from the `boost` and `less` public nonReentrant functions in TurboSafe.sol, they will revert.

Impact

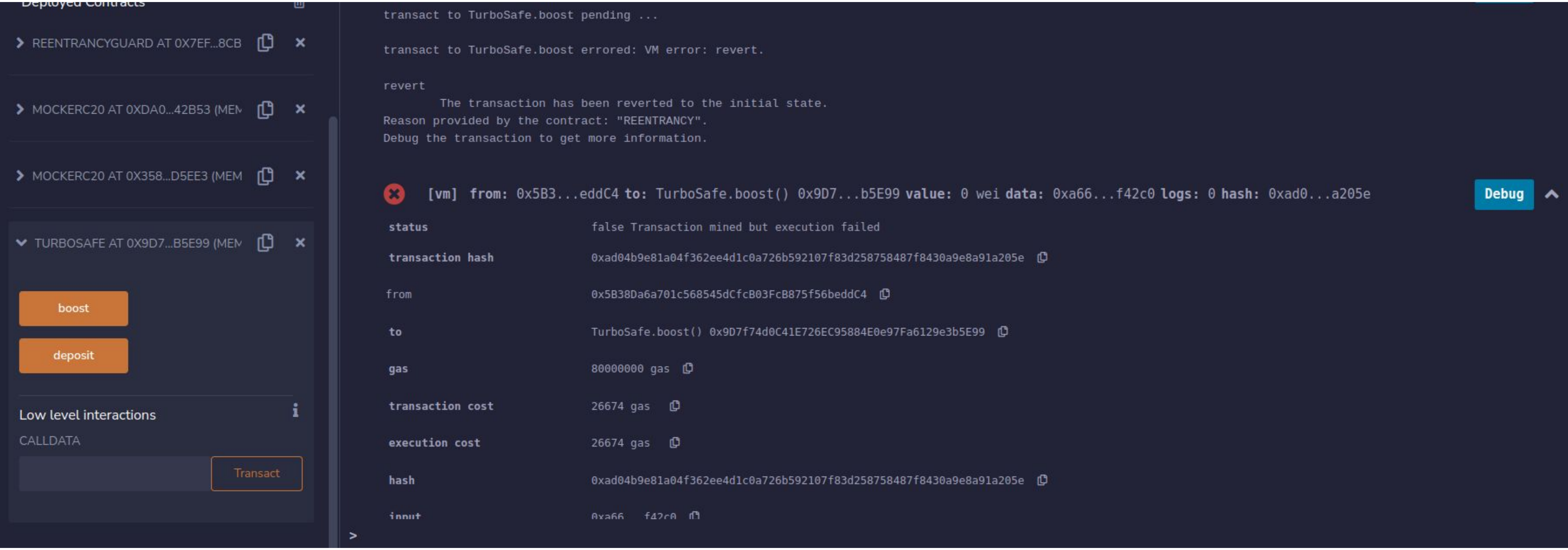
High, because the functions cannot be used in current form

Risk Breakdown

Difficulty to Exploit: Easy, just use the contract normally and experience a revert

Recommendation

Remove the nonReentrant modifier from the `beforeWithdraw` and `afterDeposit` internal functions


Deployed Contracts
REENTRANCYGUARD AT 0X7EF...8CB
MOCKERC20 AT 0XDA0...42B53 (MEM
MOCKERC20 AT 0X358...D5EE3 (MEM
TURBOSAFE AT 0X9D7...B5E99 (MEM
boost
deposit
Low level interactions
CALLDATA
Transact
Debug

```
transact to TurboSafe.boost pending ...

transact to TurboSafe.boost errored: VM error: revert.

revert
        The transaction has been reverted to the initial state.
Reason provided by the contract: "REENTRANCY".
Debug the transaction to get more information.

[vm] from: 0x5B3...eddC4 to: TurboSafe.boost() 0x9D7...b5E99 value: 0 wei data: 0xa66...f42c0 logs: 0 hash: 0xad0...a205e

status              false Transaction mined but execution failed
transaction hash    0xad04b9e81a04f362ee4d1c0a726b592107f83d258758487f8430a9e8a91a205e
from                0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                  TurboSafe.boost() 0x9D7f74d0C41E726EC95884E0e97Fa6129e3b5E99
gas                 80000000 gas
transaction cost    26674 gas
execution cost      26674 gas
hash                0xad04b9e81a04f362ee4d1c0a726b592107f83d258758487f8430a9e8a91a205e
input               0xa66...f42c0
>
```

3. Remove unnecessary nonReentrant modifiers for gas savings

The only two functions that benefit from the nonReentrant modifier are the `boost` and `slurp` functions of TurboSafe.sol. The other functions can remove the nonReentrant modifier because they either 1. do not modify state variables or 2. follow the checks-effects-interaction pattern. As a result, the ReentrancyGuard import in TurboGibber.sol can be completely removed.

Proof of concept/Steps to Reproduce

Manual testing

Impact

Gas savings

Risk Breakdown

Gas savings

Recommendation

Remove the nonReentrant modifier from all functions besides `boost` and `slurp`

4. Gas optimization in TurboClerk.sol

As mentioned in the initial presentation:

Line 108 of TurboClerk.sol

```
if (getCustomFeePercentageForSafe[safe] != 0) return getCustomFeePercentageForSafe[safe];
```

we can cache this value for gas savings. The updated code might look like

```
// Get the custom fee percentage set for the Safe

uint256 customFeePercentageForSafe = getCustomFeePercentageForSafe[safe];

  if (customFeePercentageForSafe != 0) return customFeePercentageForSafe;
```

5. Unnecessary unchecked clause

There is an unchecked clause in TurboMaster.sol around code that performs no arithmetic operations. The unchecked clause can be removed because it doesn't provide gas savings.

Proof of concept/Steps to Reproduce

Lines 252-260 of TurboMaster.sol have an unnecessary unchecked clause. The code inside the clause only sets two variables and does not benefit from unchecked

```
getTotalBoostedForVault[vault] = newTotalBoostedForVault;

getTotalBoostedAgainstCollateral[underlying] = newTotalBoostedAgainstCollateral;
```

Impact

Clean code

Recommendation

Remove unnecessary unchecked clause

6. Max contract size exceeded

Proof of concept/Steps to Reproduce

Upon compile:

```
Warning: Contract code size is 24977 bytes and exceeds 24576 bytes
```

Impact

Max contract size limit is 24KB, must be fixed else the contract will not deploy.

Recommendation

Removing various pieces of code, optimizing, or adopting the Diamond Standard

7. Comment typo

The word "Safe" should be added to the end of this comment in TurboSafe.sol line 243

```
// Compute what percentage of the interest earned will go back to the
```

For more info, please see:

https://github.com/xBalbinus/tribeaudit