


DeFi Saver

Date	March 2021
Lead Auditor	Shayan Eskandari
Co-auditors	David Oz Kashi
Download	PDF 

1 Executive Summary

This report presents the results of our engagement with **DeFi Saver** to review DeFi Saver V3 architecture, which is the new version of smart contracts that are used for their dashboard.

The review was conducted over two weeks, from **March 22, 2021** to **April 2, 2021** by **Shayan Eskandari** and **David Oz Kashi**. A total of 20 person-days were spent.

2 Scope

Our review focused on the commit hash `cb29669a84c2d6fffaf2231c0938eb407c060919`. The list of files in scope can be found in the [Appendix](#). Note that functionalities regarding *Strategy* and *Subscriptions* were not in the scope of this audit.

3 System Overview

DeFi Saver acts as a proxy (dashboard) for users to interact with DeFi protocols. Users can chain actions (create recipes) and run multiple actions in one transaction.

Here is an overview of the DeFi Saver smart contract system:

4 Security Specification

This section describes, **from a security perspective**, the expected behavior of the system under audit. It is not a substitute for documentation.

4.1 Actors

The relevant actors are listed below with their respective abilities:

- Owner
 - Add new contracts to the DeFi Saver system (Registry)
 - Swap any contract in the registry.
 - Note that in the code base there is a `WaitPeriod` defined for changes in the contracts. However owner can set this wait time to 0 and consequently change the modules/contracts in DeFi Saver registry.
 - Add or Remove wrapper contracts. Wrappers are used to interact with other DeFi protocols, such as 0x, Uniswap, etc
 - Add and change the proxies used in the system
 - Change Fee wallet address
 - Set 0x Addresses
- Admin
 - `withdrawStuckFunds` – Based on the design no user funds may be kept in DeFi Saver contracts. Although if by accident that happens, Admin will be able to withdraw them to their own address

- User

4.2 Trust Model

In any system, it's important to identify what trust is expected/required between various actors. For this audit, we established the following trust model:

- Many functionalities of DeFi Saver dashboard, uses constructed *CallData* to execute different actions in the smart contract systems, hence **the UI has control over the function executed in the smart contracts**. This is essential to the security of the system. Note that due to the way the transactions are crafted on the front-end, it's not possible for the wallet (e.g. MetaMask) to show the user the exact actions that the transaction will be taking (User sees a blob of data), this increases the risk of the attack going unnoticed until after the fact. The attack can be done with simple change in the outgoing address at the end of the recipe or more sophisticated crafted transaction to steal user funds (e.g. Approve Tokens for attackers address).
- As mentioned in the Actors section, owner has the ability to change the underlying contracts without any waiting period.
- There is no allowlist for tokens that are supported in DeFi Saver. Although implicitly the allowlist for other DeFi platforms are applied when interacting with that specific protocol (e.g tokens supported by Aave when interactive with Aave), but still a maliciously-crafted token implementation can be used within DeFi Saver platform.
- The usage of the `DSProxy` pattern provides an environment that is protected from outsider access by design.

5 Findings

Each issue has an assigned severity:

- `Minor` issues are subjective in nature. They are typically suggestions around best

practices or readability. Code maintainers should use their own judgment as to whether to address such issues.

- **Medium** issues are objective in nature but are not security vulnerabilities. These should be addressed unless there is a clear reason not to.
- **Major** issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- **Critical** issues are directly exploitable security vulnerabilities that need to be fixed.

5.1 Random task execution **Critical** **Fixed**

Resolution

Fixed in [DecenterApps/defisaver-v3-contracts@478e9cd](#) by adding `ReentrancyGuard` to the `executeOperation` function.

Description

In a scenario where user takes a flash loan, `_parseFLAndExecute()` gives the flash loan wrapper contract (`FLAaveV2`, `FLDyDx`) the permission to execute functions on behalf of the user's `DSProxy`. This execution permission is revoked only after the entire recipe execution is finished, which means that in case that any of the external calls along the recipe execution is malicious, it might call `executeAction()` back and inject any task it wishes (e.g. take user's funds out, drain approved tokens, etc)

Examples

`code/contracts/actions/flashloan/FLAaveV2.sol:L105-L136`

```

function executeOperation(
    address[] memory _assets,
    uint256[] memory _amounts,
    uint256[] memory _fees,
    address _initiator,
    bytes memory _params
) public returns (bool) {
    require(msg.sender == AAVE_LENDING_POOL, ERR_ONLY_AAVE_CALLER);
    require(_initiator == address(this), ERR_SAME_CALLER);

    (Task memory currTask, address proxy) = abi.decode(_params, (Task, address));

    // Send FL amounts to user proxy
    for (uint256 i = 0; i < _assets.length; ++i) {
        _assets[i].withdrawTokens(proxy, _amounts[i]);
    }

    address payable taskExecutor = payable(registry.getAddr(TASK_EXECUTOR_ID));

    // call Action execution
    IDSPProxy(proxy).execute{value: address(this).balance}(
        taskExecutor,
        abi.encodeWithSelector(CALLBACK_SELECTOR, currTask, bytes32(_amounts[0] + _fees[0]))
    );

    // return FL
    for (uint256 i = 0; i < _assets.length; i++) {
        _assets[i].approveToken(address(AAVE_LENDING_POOL), _amounts[i] + _fees[i]);
    }

    return true;
}

```

Recommendation

A reentrancy guard (mutex) that covers the entire content of `FLAAaveV2.executeOperation` / `FLDyDx.callFunction` should be used to prevent such attack.

5.2 Tokens with more than 18 decimal points will cause issues

Major

Fixed

Resolution

Fixed in [DecenterApps/defisaver-v3-contracts@de22007](#) by using `SafeMath.sub` to revert on tokens with Decimal > 18

Description

It is assumed that the maximum number of decimals for each token is 18. However uncommon, but it is possible to have tokens with more than 18 decimals, as an Example [YAMv2](#) has 24 decimals. This can result in broken code flow and unpredictable outcomes (e.g. an underflow will result with really high rates).

Examples

- contracts/exchangeV3/wrappersV3/KyberWrapperV3.sol

```
function getSellRate(address _srcAddr, address _destAddr, uint _srcAmount, bytes memory) public returns (rate, ) {
    (rate, ) = KyberNetworkProxyInterface(KYBER_INTERFACE)
        .getExpectedRate(IERC20(_srcAddr), IERC20(_destAddr), _srcAmount);

    // multiply with decimal difference in src token
    rate = rate * (10**(18 - getDecimals(_srcAddr)));
    // divide with decimal difference in dest token
    rate = rate / (10**(18 - getDecimals(_destAddr)));
}
```

- code/contracts/views/AaveView.sol : also used in getLoanData()

Recommendation

Make sure the code won't fail in case the token's decimals is more than 18.

5.3 Error codes of Compound's Comptroller.enterMarket, Comptroller.exitMarket are not checked Major Fixed

Resolution

Fixed in [DecenterApps/defisaver-v3-contracts@7075e49](#) by reverting in the case the return value is non zero.

Description

Compound's `enterMarket/exitMarket` functions return an error code instead of reverting in case of failure. DeFi Saver smart contracts never check for the error codes returned from Compound smart contracts, although the code flow might revert due to unavailability of the CTokens, however early on checks for Compound errors are suggested.

Examples

code/contracts/actions/compound/helpers/CompHelper.sol:L26-L37

```
function enterMarket(address _cTokenAddr) public {
    address[] memory markets = new address[](1);
    markets[0] = _cTokenAddr;

    IComptroller(COMPTROLLER_ADDR).enterMarkets(markets);
}

/// @notice Exits the Compound market
/// @param _cTokenAddr CToken address of the token
function exitMarket(address _cTokenAddr) public {
    IComptroller(COMPTROLLER_ADDR).exitMarket(_cTokenAddr);
}
```

Recommendation

Caller contract should revert in case the error code is not 0.

5.4 Reversed order of parameters in allowance function call

Medium

Fixed

Resolution

Fixed in [DecenterApps/defisaver-v3-contracts@8b5657b](#) by swapping the order of function call parameters.

Description

When trying to pull the maximum amount of tokens from an approver to the allowed spender, the parameters that are used for the `allowance` function call are not in the same order that is used later in the call to `safeTransferFrom`.

Examples

`code/contracts/utils/TokenUtils.sol:L26-L44`

```
function pullTokens(
    address _token,
    address _from,
    uint256 _amount
) internal returns (uint256) {
    // handle max uint amount
    if (_amount == type(uint256).max) {
        uint256 allowance = IERC20(_token).allowance(address(this), _from);
        uint256 balance = getBalance(_token, _from);

        _amount = (balance > allowance) ? allowance : balance;
    }

    if (_from != address(0) && _from != address(this) && _token != ETH_ADDR && _amount != 0) {
        IERC20(_token).safeTransferFrom(_from, address(this), _amount);
    }

    return _amount;
}
```

Recommendation

Reverse the order of parameters in `allowance` function call to fit the order that is in the `safeTransferFrom` function call.

5.5 Full test suite is recommended

Medium

Pending

Description

The test suite at this stage is not complete and many of the tests fail to execute. For

complicated systems such as DeFi Saver, which uses many different modules and interacts with different DeFi protocols, it is crucial to have a full test coverage that includes the edge cases and failed scenarios. Especially this helps with safer future development and upgrading each modules.

As we've seen in some smart contract incidents, a complete test suite can prevent issues that might be hard to find with manual reviews.

Some issues such as [issue 5.4](#) could be caught by a full coverage test suite.

5.6 Kyber getRates code is unclear Minor

Description

In `contracts/exchangeV3/wrappersV3/KyberWrapperV3.sol` the function names don't reflect their true functionalities, and the code uses some undocumented assumptions.

Examples

- `getSellRate` can be converted into one function to get the rates, which then for buy or sell can swap input and output tokens
- `getBuyRate` uses a 3% slippage that is not documented.

```
function getSellRate(address _srcAddr, address _destAddr, uint _srcAmount, bytes memory) public returns (uint256 rate) {
    (rate, ) = KyberNetworkProxyInterface(KYBER_INTERFACE)
        .getExpectedRate(IERC20(_srcAddr), IERC20(_destAddr), _srcAmount);

    // multiply with decimal difference in src token
    rate = rate * (10**(18 - getDecimals(_srcAddr)));
    // divide with decimal difference in dest token
    rate = rate / (10**(18 - getDecimals(_destAddr)));
}

/// @notice Return a rate for which we can buy an amount of tokens
/// @param _srcAddr From token
/// @param _destAddr To token
/// @param _destAmount To amount
/// @return rate Rate
function getBuyRate(address _srcAddr, address _destAddr, uint _destAmount, bytes memory _additionalData) public returns (uint256 rate) {
    uint256 srcRate = getSellRate(_destAddr, _srcAddr, _destAmount, _additionalData);
    uint256 srcAmount = wmul(srcRate, _destAmount);

    rate = getSellRate(_srcAddr, _destAddr, srcAmount, _additionalData);

    // increase rate by 3% too account for inaccuracy between sell/buy conversion
    rate = rate + (rate / 30);
}
```

Recommendation

Refactoring the code to separate getting rate functionality with `getSellRate` and `getBuyRate`. Explicitly document any assumptions in the code (slippage, etc)

5.7 Missing check in `IOffchainWrapper.takeOrder` implementation

Description

`IOffchainWrapper.takeOrder` wraps an external call that is supposed to perform a token swap. As for the two different implementations `ZeroxWrapper` and `ScpWrapper` this function validates that the destination token balance after the swap is greater than the value before. However, it is not sufficient, and the user-provided minimum amount for swap should be taken in consideration as well. Besides, the external contract should not be trusted upon, and `SafeMath` should be used for the subtraction operation.

Examples

`code/contracts/exchangeV3/offchainWrappersV3/ZeroxWrapper.sol:L42-L50`

```
uint256 tokensBefore = _exData.destAddr.getBalance(address(this));
(success, ) = _exData.offchainData.exchangeAddr.call{value: _exData.offchainData.protocolFee}(_exData);
uint256 tokensSwaped = 0;

if (success) {
    // get the current balance of the swaped tokens
    tokensSwaped = _exData.destAddr.getBalance(address(this)) - tokensBefore;
    require(tokensSwaped > 0, ERR_TOKENS_SWAPED_ZERO);
}
```

`code/contracts/exchangeV3/offchainWrappersV3/ScpWrapper.sol:L43-L51`

```
uint256 tokensBefore = _exData.destAddr.getBalance(address(this));
(success, ) = _exData.offchainData.exchangeAddr.call{value: _exData.offchainData.protocolFee}(_exData);
uint256 tokensSwaped = 0;

if (success) {
    // get the current balance of the swaped tokens
    tokensSwaped = _exData.destAddr.getBalance(address(this)) - tokensBefore;
    require(tokensSwaped > 0, ERR_TOKENS_SWAPED_ZERO);
}
```

5.8 Unused code present in the codebase Minor

Resolution

Some of the unused code were removed in [DecenterApps/defisaver-v3-contracts@61b0c09](#).

Description

There are a few instances of unused code (dead code) in the code base, that is suggested to be removed.

Examples

- `DFSExchange.sol` contract is not used
- `/contracts/utils/ZrxAllowlist.sol` these functions are not used in the codebase:
 - `nonPayableAddr` mapping
 - `addNonPayableAddr()`
 - `removeNonPayableAddr()`
 - `isNonPayableAddr()`
- `DSPProxy.execute(bytes memory _code, bytes memory _data)` is not intended to be used.

There might be more instances of unused code in the codebase.

5.9 Return values not used for `DFSExchangeCore.onChainSwap`

Minor

Description

Return values from `DFSExchangeCore.onChainSwap` are not used.

Examples

`code/contracts/exchangeV3/DFSExchangeCore.sol:L37-L73`

```

function _sell(ExchangeData memory exData) internal returns (address, uint256) {
    uint256 amountWithoutFee = exData.srcAmount;
    address wrapper = exData.offchainData.wrapper;
    bool offChainSwapSuccess;

    uint256 destBalanceBefore = exData.destAddr.getBalance(address(this));

    // Takes DFS exchange fee
    exData.srcAmount -= getFee(
        exData.srcAmount,
        exData.user,
        exData.srcAddr,
        exData.dfsFeeDivider
    );

    // Try 0x first and then fallback on specific wrapper
    if (exData.offchainData.price > 0) {
        (offChainSwapSuccess, ) = offChainSwap(exData, ExchangeActionType.SELL);
    }

    // fallback to desired wrapper if 0x failed
    if (!offChainSwapSuccess) {
        onChainSwap(exData, ExchangeActionType.SELL);
        wrapper = exData.wrapper;
    }

    uint256 destBalanceAfter = exData.destAddr.getBalance(address(this));
    uint256 amountBought = sub(destBalanceAfter, destBalanceBefore);

    // check slippage
    require(amountBought >= wmul(exData.minPrice, exData.srcAmount), ERR_SLIPPAGE_HIT);

    // revert back exData changes to keep it consistent
    exData.srcAmount = amountWithoutFee;

    return (wrapper, amountBought);
}

```

code/contracts/exchangeV3/DFSExchangeCore.sol:L79-L117

```

function _buy(ExchangeData memory exData) internal returns (address, uint256) {
    require(exData.destAmount != 0, ERR_DEST_AMOUNT_MISSING);

    uint256 amountWithoutFee = exData.srcAmount;
    address wrapper = exData.offchainData.wrapper;
    bool offChainSwapSuccess;

    uint256 destBalanceBefore = exData.destAddr.getBalance(address(this));

    // Takes DFS exchange fee
    exData.srcAmount -= getFee(
        exData.srcAmount,
        exData.user,
        exData.srcAddr,
        exData.dfsFeeDivider
    );

    // Try 0x first and then fallback on specific wrapper
    if (exData.offchainData.price > 0) {
        (offChainSwapSuccess, ) = offChainSwap(exData, ExchangeActionType.BUY);
    }

    // fallback to desired wrapper if 0x failed
    if (!offChainSwapSuccess) {
        onChainSwap(exData, ExchangeActionType.BUY);
        wrapper = exData.wrapper;
    }

    uint256 destBalanceAfter = exData.destAddr.getBalance(address(this));
    uint256 amountBought = sub(destBalanceAfter, destBalanceBefore);

    // check slippage
    require(amountBought >= exData.destAmount, ERR_SLIPPAGE_HIT);

    // revert back exData changes to keep it consistent
    exData.srcAmount = amountWithoutFee;

    return (wrapper, amountBought);
}

```

Recommendation

The return value can be used for verification of the swap or used in the event data.

5.10 Return value is not used for `TokenUtils.withdrawTokens`

Minor

Fixed

Resolution

Fixed in [DecenterApps/defisaver-v3-contracts@37dabff](#) by storing the return value locally and use its value throughout the execution.

Description

The return value of `TokenUtils.withdrawTokens` which represents the actual amount of tokens that were transferred is never used throughout the repository. This might cause discrepancy in the case where the original value of `_amount` was `type(uint256).max`.

Examples

`code/contracts/actions/aave/AaveBorrow.sol:L70-L97`

```
function _borrow(
    address _market,
    address _tokenAddr,
    uint256 _amount,
    uint256 _rateMode,
    address _to,
    address _onBehalf
) internal returns (uint256) {
    ILendingPoolV2 lendingPool = getLendingPool(_market);

    // defaults to onBehalf of proxy
    if (_onBehalf == address(0)) {
        _onBehalf = address(this);
    }

    lendingPool.borrow(_tokenAddr, _amount, _rateMode, AAVE_REFERRAL_CODE, _onBehalf);

    _tokenAddr.withdrawTokens(_to, _amount);

    logger.Log(
        address(this),
        msg.sender,
        "AaveBorrow",
        abi.encode(_market, _tokenAddr, _amount, _rateMode, _to, _onBehalf)
    );

    return _amount;
}
```

`code/contracts/utils/TokenUtils.sol:L46-L53`

```
function withdrawTokens(
    address _token,
    address _to,
    uint256 _amount
) internal returns (uint256) {
    if (_amount == type(uint256).max) {
        _amount = getBalance(_token, address(this));
    }
}
```

Recommendation

The return value can be used to validate the withdrawal or used in the event emitted.

5.11 Missing access control for DefiSaverLogger.Log

Description

`DefiSaverLogger` is used as a logging aggregator within the entire dapp, but anyone can create

logs.

Examples

code/contracts/Utils/DefisaverLogger.sol:L14-L21

```
function Log(  
    address _contract,  
    address _caller,  
    string memory _logName,  
    bytes memory _data  
) public {  
    emit LogEvent(_contract, _caller, _logName, _data);  
}
```

6 Recommendations

6.1 Use a single file for all system-wide constants

Description

There are many addresses and constants used in the system. It is suggested to put the most used ones in one file (e.g. `constants.sol`) and use inheritance to access these values. This will help with the readability and easier maintenance for future changes. As some of these hardcoded values are admin addresses, this also helps with any possible incident response.

Examples

Logger:

- DFSRegistry
- TaskExecutor
- ActionBase

```
DefisaverLogger public constant logger = DefisaverLogger(  
    0x5c55B921f590a89C1Ebe84dF170E655a82b62126  
);
```

Admin Vault:

- AdminAuth

```
AdminVault public constant adminVault = AdminVault(0xCCf3d848e08b94478Ed8f46fFea3008faF581fD);
```

REGISTRY_ADDR

- SubscriptionProxy
- StrategyExecutor
- TaskExecutor
- ActionBase

```
address public constant REGISTRY_ADDR = 0xB0e1682D17A96E8551191c089673346dF7e1D467;
```

Any other `constant` in the system also can be moved to this contract.

Recommendation

Use `constants.sol` and import this file in the contracts that require access to these values. This is just a recommendation, as discussed with the team, on some use cases this might result in higher gas usage on deployment.

6.2 Code quality & Styling

Description

Here are some examples that the code style does not follow the best practices:

Examples

- Public/external function names should not be prefixed with `_`

`code/contracts/core/TaskExecutor.sol:L56`

```
function _executeActionsFromFL(Task memory _currTask, bytes32 _flAmount) public payable {
```

- Function parameters are being overridden

`code/contracts/exchangeV3/DFSExchange.sol:L24-L37`

```
function sell(ExchangeData memory exData, address payable _user) public payable {  
  
    exData.dfsFeeDivider = SERVICE_FEE;  
    exData.user = _user;  
  
    // Perform the exchange  
    (address wrapper, uint destAmount) = _sell(exData);  
  
    // send back any leftover ether or tokens  
    sendLeftover(exData.srcAddr, exData.destAddr, _user);  
  
    // log the event  
    logger.Log(address(this), msg.sender, "ExchangeSell", abi.encode(wrapper, exData.srcAddr, exData.c  
}
```

- `MAX_SERVICE_FEE` should be `MIN_SERVICE_FEE`

`code/contracts/utils/Discount.sol:L28-L33`

```
function setServiceFee(address _user, uint256 _fee) public {
    require(msg.sender == owner, "Only owner");
    require(_fee >= MAX_SERVICE_FEE || _fee == 0, "Wrong fee value");

    serviceFees[_user] = CustomServiceFee({active: true, amount: _fee});
}
```

- Functions with a `get` prefix should not modify state

code/contracts/exchangeV3/DFSExchangeCore.sol:L182-L206

```
function getFee(
    uint256 _amount,
    address _user,
    address _token,
    uint256 _dfsFeeDivider
) internal returns (uint256 feeAmount) {
    if (_dfsFeeDivider != 0 && Discount(DISCOUNT_ADDRESS).isCustomFeeSet(_user)) {
        _dfsFeeDivider = Discount(DISCOUNT_ADDRESS).getCustomServiceFee(_user);
    }

    if (_dfsFeeDivider == 0) {
        feeAmount = 0;
    } else {
        feeAmount = _amount / _dfsFeeDivider;

        // fee can't go over 10% of the whole amount
        if (feeAmount > (_amount / 10)) {
            feeAmount = _amount / 10;
        }

        address walletAddr = feeRecipient.getFeeAddr();

        _token.withdrawTokens(walletAddr, feeAmount);
    }
}
```

- Protocol fee value should be validated against `msg.value` and not against contract's balance

code/contracts/exchangeV3/offchainWrappersV3/ZeroxWrapper.sol:L25-L31

```
function takeOrder(
    ExchangeData memory _exData,
    ExchangeActionType _type
) override public payable returns (bool success, uint256) {
    // check that contract have enough balance for exchange and protocol fee
    require(_exData.srcAddr.getBalance(address(this)) >= _exData.srcAmount, ERR_SRC_AMOUNT);
    require(TokenUtils.ETH_ADDR.getBalance(address(this)) >= _exData.offchainData.protocolFee, ERR_PRC
```

- Remove deprecation warning (originated in OpenZeppelin's implementation) in comment, as the issue has been solved

code/contracts/utils/SafeERC20.sol:L33-L44

```

/**
 * @dev Deprecated. This function has issues similar to the ones found in
 * {ERC20-approve}, and its usage is discouraged.
 */
function safeApprove(
    IERC20 token,
    address spender,
    uint256 value
) internal {
    _callOptionalReturn(token, abi.encodeWithSelector(token.approve.selector, spender, 0));
    _callOptionalReturn(token, abi.encodeWithSelector(token.approve.selector, spender, value));
}

```

- Typo `RECIPIE_FEE` instead of `RECIPE_FEE`

code/contracts/actions/exchange/DfsSell.sol:L15

```

uint internal constant RECIPIE_FEE = 400;

```

- Code duplication : `sendLeftOver` is identical both in `UniswapWrapperV3` and in `KyberWrapperV3`, and thus can be shared in a base class.

code/contracts/exchangeV3/wrappersV3/KyberWrapperV3.sol:L127-L133

```

function sendLeftOver(address _srcAddr) internal {
    msg.sender.transfer(address(this).balance);

    if (_srcAddr != KYBER_ETH_ADDRESS) {
        IERC20(_srcAddr).safeTransfer(msg.sender, IERC20(_srcAddr).balanceOf(address(this)));
    }
}

```

- Code duplication : `sliceUint` function is identical both in `DFSExchangeHelper` and in `DFSPrices`
- `DFSPricesV3.getBestPrice`, `DFSPricesV3.getExpectedRate` should be view functions
- Fix the code comments from `User borrows tokens to` to `User borrows tokens from`

code/contracts/actions/aave/AaveBorrow.sol:L63-L77


```

/// @notice User borrows tokens to the Aave protocol
/// @param _market Address provider for specific market
/// @param _tokenAddr The address of the token to be borrowed
/// @param _amount Amount of tokens to be borrowed
/// @param _rateMode Send 1 for stable rate and 2 for variable
/// @param _to The address we are sending the borrowed tokens to
/// @param _onBehalf From what user we are borrow the tokens, defaults to proxy
function _borrow(
    address _market,
    address _tokenAddr,
    uint256 _amount,
    uint256 _rateMode,
    address _to,
    address _onBehalf
) internal returns (uint256) {

```

code/contracts/actions/compound/CompBorrow.sol:L51-L59

```

/// @notice User borrows tokens to the Compound protocol
/// @param _cTokenAddr Address of the cToken we are borrowing
/// @param _amount Amount of tokens to be borrowed
/// @param _to The address we are sending the borrowed tokens to
function _borrow(
    address _cTokenAddr,
    uint256 _amount,
    address _to
) internal returns (uint256) {

```

- `IEExchangeV3.sell`, `IEExchangeV3.buy` should not be payable
- `TaskExecutor._executeAction` should not forward contract's balance within the `IDSPProxy.execute` call, as the funds are being sent to the same contract.

code/contracts/core/TaskExecutor.sol:L90-L105

```

function _executeAction(
    Task memory _currTask,
    uint256 _index,
    bytes32[] memory _returnValues
) internal returns (bytes32 response) {
    response = IDSPProxy(address(this)).execute{value: address(this).balance}(
        registry.getAddr(_currTask.actionIds[_index]),
        abi.encodeWithSignature(
            "executeAction(bytes[],bytes[],uint8[],bytes32[])",
            _currTask.callData[_index],
            _currTask.subData[_index],
            _currTask.paramMapping[_index],
            _returnValues
        )
    );
}

```

- Unsafe arithmetic operations

code/contracts/actions/compound/CompClaim.sol:L73

```
uint256 compClaimed = compBalanceAfter - compBalanceBefore;
```

code/contracts/actions/compound/CompWithdraw.sol:L84

```
_amount = tokenBalanceAfter - tokenBalanceBefore;
```

code/contracts/actions/uniswap/UniSupply.sol:L82-L83

```
_uniData.tokenA.withdrawTokens(_uniData.to, (_uniData.amountADesired - amountA));  
_uniData.tokenB.withdrawTokens(_uniData.to, (_uniData.amountBDesired - amountB));
```

code/contracts/actions/flashloan/FLAaveV2.sol:L125-L133

```
IDSPProxy(proxy).execute{value: address(this).balance}(  
    taskExecutor,  
    abi.encodeWithSelector(CALLBACK_SELECTOR, currTask, bytes32(_amounts[0] + _fees[0]))  
);  
  
// return FL  
for (uint256 i = 0; i < _assets.length; i++) {  
    _assets[i].approveToken(address(AAVE_LENDING_POOL), _amounts[i] + _fees[i]);  
}
```

code/contracts/exchangeV3/DFSEExchangeCore.sol:L45

```
exData.srcAmount -= getFee(
```

code/contracts/exchangeV3/offchainWrappersV3/ZeroxWrapper.sol:L48

```
tokensSwaped = _exData.destAddr.getBalance(address(this)) - tokensBefore;
```

6.3 Gas optimization

Description

Use `address(this)` instead of external call for registry when possible.

Examples

code/contracts/actions/flashloan/FLAaveV2.sol:L82-L102

```

function _fLAaveV2(FLAaveV2Data memory _flData, bytes memory _params) internal returns (uint) {

    ILendingPoolV2(AAVE_LENDING_POOL).flashLoan(
        payable(registry.getAddr(FL_AAVE_V2_ID)),
        _flData.tokens,
        _flData.amounts,
        _flData.modes,
        _flData.onBehalfOf,
        _params,
        AAVE_REFERRAL_CODE
    );

    logger.Log(
        address(this),
        msg.sender,
        "FLAaveV2",
        abi.encode(_flData.tokens, _flData.amounts, _flData.modes, _flData.onBehalfOf)
    );

    return _flData.amounts[0];
}

```

code/contracts/actions/flashloan/dydx/FLDyDx.sol:L76-L107

```

function _fLDyDx(
    uint256 _amount,
    address _token,
    bytes memory _data
) internal returns (uint256) {

    address payable receiver = payable(registry.getAddr(FL_DYDX_ID));

    ISoloMargin solo = ISoloMargin(SOLO_MARGIN_ADDRESS);

    // Get marketId from token address
    uint256 marketId = _getMarketIdFromTokenAddress(SOLO_MARGIN_ADDRESS, _token);

    uint256 repayAmount = _getRepaymentAmountInternal(_amount);

    IERC20(_token).safeApprove(SOLO_MARGIN_ADDRESS, repayAmount);

    Actions.ActionArgs[] memory operations = new Actions.ActionArgs[](3);

    operations[0] = _getWithdrawAction(marketId, _amount, receiver);
    operations[1] = _getCallAction(_data, receiver);
    operations[2] = _getDepositAction(marketId, repayAmount, address(this));

    Account.Info[] memory accountInfos = new Account.Info[](1);
    accountInfos[0] = _getAccountInfo();

    solo.operate(accountInfos, operations);

    logger.Log(address(this), msg.sender, "FLDyDx", abi.encode(_amount, _token));

    return _amount;
}

```

Appendix 1 – Files in Scope

This audit covered the following files:

File Name	SHA-1 Hash
contracts/auth/AdminVault.sol	0bcc845ec8e2d927ca7ade0ab31471083cd798a5
contracts/auth/AdminAuth.sol	ebe4c9219e473983df73569bed7b84008d0b0251
contracts/auth/ProxyPermission.sol	cee91dbdd730811837a25ee92868e090ffb5220e
contracts/core/TaskExecutor.sol	1a456a05404bb5b9bffd2ee8d726e62b777c644
contracts/core/DFSRegistry.sol	19f2678b2d7795f2d579644db14f0e2686792c1c
contracts/utils/SafeERC20.sol	9411f7bd95ba807e0a125219497b9b8be42f0446
contracts/utils/Discount.sol	d9495bfb48bf8251143a1a30bd845593e9488e11
contracts/utils/FeeRecipient.sol	eae0bf0c1b0abc1250be6fa49b96924226669d2e
contracts/utils/FLFeeFaucet.sol	6ff7e59b32e184ff66fdd1c27c5fdc76f721564d
contracts/utils/Exponential.sol	aa4098007240494f375dd3533b5d02d5bdd4d8b4
contracts/utils/SafeMath.sol	4381feeda6079de2addc7e267657a4ef2658dc6c
contracts/utils/Address.sol	24762c686cd3cf849197cd912858226a774b5e6e
contracts/utils/ZrxAllowlist.sol	f20a47bb1be3272d5e8954a040c62a89612b5dfc
contracts/utils/DFSProxyRegistry.sol	d541d59864694dd6b19187c9e4231286afce961c
contracts/utils/DefisaverLogger.sol	d2362e116c43593168d2c00adb7472a916230d63
contracts/utils/TokenUtils.sol	492839d9ac138304af554933db621c2d5bdf8550
contracts/utils/CarefulMath.sol	327ed2e92a98e57759da3e8e41e3c28a6128c169
	10cccd7c227c235f86c7cbb5f66b2cf274f200

File Name	SHA-1 Hash
contracts/exchangeV3/DFSExchangeHelper.sol	19ead7a237a53180e7cbb3100b2c137412300534
contracts/exchangeV3/wrappersV3/KyberWrapperV3.sol	294efb5079e81052bcc682062dae28d45096b989
contracts/exchangeV3/offchainWrappersV3/ScpWrapper.sol	1c3c20a94a49e03829c701e1fa0ae9fb75895449
contracts/exchangeV3/wrappersV3/UniswapWrapperV3.sol	f845341af105f8dc456a03ada15b58119dbd38e0
contracts/exchangeV3/DFSExchangeCore.sol	cf1bb7f3e692300404ac8cde83f9c1fb50e2b594
contracts/exchangeV3/DFSExchangeData.sol	50cf9d3288b28032878d737040353fc97407976b
contracts/exchangeV3/DFSExchange.sol	424120fc97700cb4046928ec99db3b789bf86199
contracts/exchangeV3/DFSPricesV3.sol	8d9ea0ac0bd63af56cc53d8f38fdad10f8854c6a
contracts/exchangeV3/SaverExchangeRegistry.sol	79708bca57219b8178a404b9afe18d6a143c648a
contracts/exchangeV3/offchainWrappersV3/ZeroxWrapper.sol	bf63611a717b7991fc8d23682fc0688afb8a906e
contracts/actions/utils/UnwrapEth.sol	90ff17831ee096c413af2cae007139fc3730e5cb
contracts/actions/utils/PullToken.sol	4692b288725331186447185cbb5eb1b32485ad10
contracts/actions/utils/SumInputs.sol	edd5622e65882938d9c4f3425b049101c659e70d
contracts/actions/exchange/DfsSell.sol	d6e7df7045af542fea38a8a95e4dd443b2be0f64
contracts/actions/flashloan/FLAaveV2.sol	f5383042870d142684a5a78c36046a37b0a73b84
contracts/actions/ActionBase.sol	86e3425f0160a86d6fbe3b7293fd2220d160721a
contracts/actions/exchange/DFSBuy.sol	57d296223fb587b92dd1bb61aa3d7c931ae41433
contracts/actions/uniswap/UniWithdraw.sol	6358c7ceec12b52a7d75bd37a2e8ac90dca95da0
contracts/actions/uniswap/UniSupply.sol	2d670ef9f9a4cc83bdef2ac54bef88a3cfcd efcc
contracts/actions/utils/WrapEth.sol	79ec4b231dac21feedf28d2e8cf2bae8e7a87fe4

File Name	SHA-1 Hash
contracts/actions/flashloan/dydx/DydxFlashLoanBase.sol	9a40f771ef6397f97e68335e26f6283403371934
contracts/actions/compound/CompWithdraw.sol	dfadbb93dbb3f0ace1d5ee2eb2247e7ed0d85472
contracts/actions/flashloan/dydx/FLDyDx.sol	888d35f3a97a003737f93e360ec46ba11facb93b
contracts/actions/utils/SendToken.sol	1aa1d4b8e241e0ffc62dfef14b900f614ab28461
contracts/actions/aave/AaveSupply.sol	55a00932fd4db3e6778df0cafbfeaa0871db6fc6
contracts/actions/mcd/McdMerge.sol	b04431602f8a4429e38d1a486eb37bb8eab164a1
contracts/actions/aave/AaveWithdraw.sol	65c8dd23d6076edb4b13194c165c86563408f50e
contracts/actions/mcd/McdPayback.sol	b5fe10f2913e6adc4c56e1397fe819574a814a3f
contracts/actions/aave/AaveBorrow.sol	1b43deed1be0a5aee5ab7febd0e85e24f77efe99d
contracts/actions/compound/CompBorrow.sol	170c9e7b40601fcd0389f91803675eb6fbbeb616
contracts/actions/compound/CompSupply.sol	30c45f093217173853b8a09cb97505f8a53eb0ce
contracts/actions/aave/AavePayback.sol	a8f56e6bdbd17c6131edc2b7e81c910ba821af03
contracts/actions/compound/CompPayback.sol	0ac2f420cdb306d0ae2ef9150944d9a719484fdc
contracts/actions/mcd/McdGenerate.sol	fad58d9b6de75694dffafaf1ebf7d8bba2995d243
contracts/actions/mcd/McdGive.sol	8e50727a34b43030d928fba26f3183ed594e67d1
contracts/actions/mcd/McdWithdraw.sol	a93b46770994aa900cbb3d8d366bafb6c2607959
contracts/actions/aave/helpers/AaveHelper.sol	3d299853e11e07aa19f3839c4bb57644781a1838
contracts/actions/mcd/McdSupply.sol	6f49b4069839834bf643273cbeb64a6b945775ce
contracts/actions/mcd/McdOpen.sol	dc8051109bf037f0532f565f7ca139f55bb9fcce
	e1621308e26ce316fbfdecd651cd18243e5

contracts/actions/compound/CompClaim.sol	b908	SHA-1 Hash
contracts/actions/compound/helpers/CompHelper.sol	6c2699e4dd873ca36c22c1eaaaaacfe3d2d8860f	
contracts/actions/mcd/helpers/McdHelper.sol	0ffea9e971bee2cb573ea8f6b344e2feba3fff32	

The following files were looked at to understand the overall system but are not in the scope:

File Name	SHA-1 Hash
contracts/DS/DSPProxyFactoryInterface.sol	f2366ee831535db0216213177b71c591af775dd8
contracts/DS/DSMath.sol	582030753e6d2682bb1104d175c09ce10a55e217
contracts/DS/DSGuard.sol	33221642c289acb219b35df3e5c9223ee6c59677
contracts/DS/DSAuthority.sol	af4795dfcdf38101b1d1b5543bf5baa726bbe722
contracts/DS/DSPProxy.sol	cbc0b52690cbde8540f411faa1933317116d6e9b
contracts/DS/DSNote.sol	d27a91dd0123f3793ec6d437dc40bf1cb66957f5
contracts/DS/DSAAuth.sol	3df0ea67a517b9ac3d60d454a3d3fe77fb80534a

Appendix 2 – Disclosure

ConsenSys Diligence (“CD”) typically receives compensation from one or more clients (the “Clients”) for performing the analysis contained in these reports (the “Reports”). The Reports may be distributed through other means, including via ConsenSys publications and other distributions.

The Reports are not an endorsement or indictment of any particular project or team, and the Reports do not guarantee the security of any particular project. This Report does not consider, and should not be interpreted as considering or having any bearing on, the potential economics of a token, token sale or any other product, service or other asset. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty. No Report provides any warranty or representation to any Third-Party in any respect, including regarding the bugfree nature of code, the business model or proprietors of any such business model, and the legal compliance of any such business. No third party should rely on the Reports in any way, including for the purpose of making any decisions to buy or sell any token, product, service or other asset. Specifically, for the avoidance of doubt, this Report does not constitute investment advice, is not intended to be relied upon as investment advice, is not an endorsement of this project or team, and it is not a guarantee as to the absolute security of the project. CD owes no duty to any Third-Party by virtue of publishing these Reports.

PURPOSE OF REPORTS The Reports and the analysis described therein are created solely for Clients and published with their consent. The scope of our review is limited to a review of Solidity code and only the Solidity code we note as being within the scope of our review within this report. The Solidity language itself 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 Solidity that could present security risks. Cryptographic tokens are emergent technologies and carry with them high levels of technical risk and uncertainty.

CD makes the Reports available to parties other than the Clients (i.e., “third parties”) - on its website. CD hopes that by making these analyses publicly available, it can help the blockchain ecosystem develop technical best practices in this rapidly evolving area of innovation.

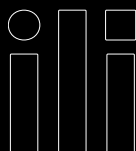
LINKS TO OTHER WEB SITES FROM THIS WEB SITE You may, through hypertext or other computer links, gain access to web sites operated by persons other than ConsenSys and CD. Such hyperlinks are provided for your reference and convenience only, and are the exclusive responsibility of such web sites’ owners. You agree that ConsenSys and CD are not responsible for the content or operation of such Web sites, and that ConsenSys and CD shall have no liability to you or any other person or entity for the use of third party Web sites. Except as described below, a hyperlink from this web Site to another web site does not imply or mean that ConsenSys and CD endorses the content on that Web site or the operator or operations of that site. You are solely responsible for determining the extent to which you may use any content at any other web sites to which you link from the Reports. ConsenSys and CD assumes no responsibility for the use of third party software on the Web Site and shall have no liability whatsoever to any person or entity for the accuracy or completeness of any outcome generated by such software.

TIMELINESS OF CONTENT The content contained in the Reports is current as of the date appearing on the Report and is subject to change without notice. Unless indicated otherwise, by ConsenSys and CD.

Request a Security Review Today

Get in touch with our team to request a quote for a smart contract audit.

[CONTACT US](#)



[AUDITS](#) [FUZZING](#) [SCRIBBLE](#) [BLOG](#) [TOOLS](#) [RESEARCH](#) [ABOUT](#) [CONTACT](#) [CAREERS](#) [PRIVACY](#) [POLICY](#)

Subscribe to Our Newsletter

Stay up-to-date on our latest offerings, tools, and the world of blockchain security.

