SMART CONTRACT AUDIT

ZOKYO.

May 21, 2021 | v. 1.0

PASS

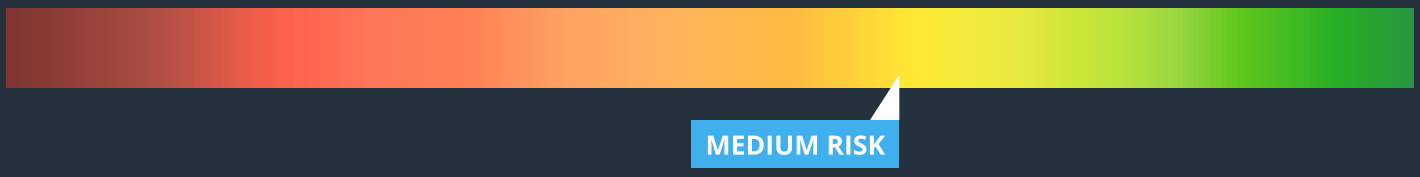
Zokyo's Security Team has concluded that this smart contract passes security qualifications to be listed on digital asset exchanges

TECHNICAL SUMMARY

This document outlines the overall security of the EqiFi smart contracts, evaluated by Zokyo's Blockchain Security team.

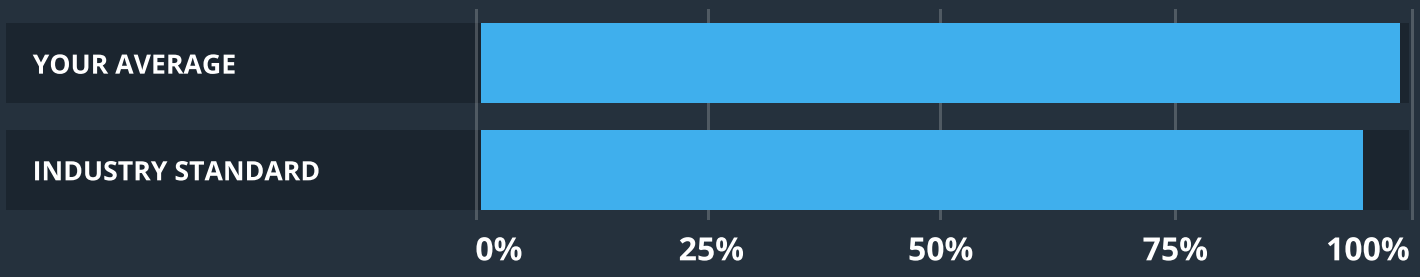
The scope of this audit was to analyze and document the EqiFi smart contract codebase for quality, security, and correctness.

Contract Status



There were 3 critical issues found during the audit.

Testable Code



Testable code is 99.04% which is above the industry standard of 95%.

It should be noted that this audit is not an endorsement of the reliability or effectiveness of the contract, rather limited to an assessment of the logic and implementation. In order to ensure a secure contract that's able to withstand the Ethereum network's fast-paced and rapidly changing environment, we at Zokyo recommend that the EqiFi team put in place a bug bounty program to encourage further and active analysis of the smart contract.

TABLE OF CONTENTS

AUDITING STRATEGY AND TECHNIQUES APPLIED

The Smart contract's source code was taken from the EqiFi repository – https://bitbucket.org/RebelDot/swap-blockchain/commits/a77d4e91c400b68e9c41bb3d0fd3ef78266894ec.

Throughout the review process, care was taken to ensure that the token contract:

- Implements and adheres to existing Token standards appropriately and effectively;
- Documentation and code comments match logic and behavior;
- Distributes tokens in a manner that matches calculations;
- Follows best practices in efficient use of gas, without unnecessary waste;
- Uses methods safe from reentrance attacks;
- Is not affected by the latest vulnerabilities;
- Whether the code meets best practices in code readability, etc.

Zokyo's Security Team has followed best practices and industry-standard techniques to verify the implementation of EqiFi smart contracts. To do so, the code is reviewed line-by-line by our smart contract developers, documenting any issues as they are discovered. Part of this work includes writing a unit test suite using the Truffle testing framework. In summary, our strategies consist largely of manual collaboration between multiple team members at each stage of the review:

1	Due diligence in assessing the overall code quality of the codebase.	3	Testing contract logic against common and uncommon attack vectors.
2	Cross-comparison with other, similar smart contracts by industry leaders.	4	Thorough, manual review of the codebase, line-by-line.

EXECUTIVE SUMMARY

There were 3 critical issues found during the audit. All the mentioned findings may have an effect only in case of specific conditions performed by the contract owner. None of the critical issues were resolved.

Generally, the contracts are well written and structured. The findings during the audit have some impact on contract performance or security, so it recommended to provide the necessary fixes. Zokyo team has left the solution for each issue and highly insists on making the edits to the existing contract.

STRUCTURE AND ORGANIZATION OF DOCUMENT

For ease of navigation, sections are arranged from most critical to least critical. Issues are tagged “Resolved” or “Unresolved” depending on whether they have been fixed or addressed. Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

Critical

The issue affects the ability of the contract to compile or operate in a significant way.

High

The issue affects the ability of the contract to compile or operate in a significant way.

Medium

The issue affects the ability of the contract to operate in a way that doesn’t significantly hinder its behavior.

Low

The issue has minimal impact on the contract’s ability to operate.

Informational

The issue has no impact on the contract’s ability to operate.

COMPLETE ANALYSIS

Ownable.sol

LOW | UNRESOLVED

It is better to use OwnableUpgradeable from Oppenzeppelin library.

Accessible.sol

MEDIUM | UNRESOLVED

Functions addOperator and removeOperator must be with modifier onlyOwner by analogy with contract Grantor.sol.

LOW | UNRESOLVED

Best practice is to call parent function init.

LOW | UNRESOLVED

Avoid extra “if”, because the owner will always be admin

```
35    function transferOwnership(address newOwner) public override onlyOwner {
36        super.transferOwnership(newOwner);
37        if (hasRole(DEFAULT_ADMIN_ROLE, _msgSender())) {
38            _setupRole(DEFAULT_ADMIN_ROLE, newOwner);
39            revokeRole(DEFAULT_ADMIN_ROLE, _msgSender());
40        }
41    }
```

Grantor.sol

CRITICAL | UNRESOLVED

The init of the ownable contract is not called in the function init. It is also necessary to determine exactly who will be the granter: the sender or the owner.

```
15      function __initializeGrantor(address owner) initializer virtual public {
16          _addGrantor(msg.sender, OWNER_UNIFORM_GRANTOR_FLAG);
17      }
```

MEDIUM | UNRESOLVED

In 42 and 49 lines we need to add an error to require.

LOW | UNRESOLVED

Inconsistency with names. Msg.sender and _msgSender are both used in the contract. Besides, in some functions, the name of the isUniformGrantor argument is exactly the same as the name of the method. A good practice is to avoid such cases.

LOW | UNRESOLVED

Extra “if” is used.

```
76      function transferOwnership(address newOwner) public override onlyOwner {
77          _removeGrantor(msg.sender);
78          super.transferOwnership(newOwner);
79          if (hasRole(DEFAULT_ADMIN_ROLE, _msgSender())) {
80              _setupRole(DEFAULT_ADMIN_ROLE, newOwner);
81              revokeRole(DEFAULT_ADMIN_ROLE, _msgSender());
82          }
83          _addGrantor(newOwner, OWNER_UNIFORM_GRANTOR_FLAG);
84      }
```

LOW | UNRESOLVED

renounceOwnership is a useless function in this case.

Recommendation:
Remove the aforementioned function.

AddressStorage.sol

LOW | UNRESOLVED

It makes no sense to move the logic into a separate contract if it is used in one place.

AddressProvider.sol

HIGH | UNRESOLVED

Ownable init is not called.

CurrencySwap.sol

HIGH | UNRESOLVED

After calling the function to remove the provider liquidation, only the value is removed, not the array element. Because of this, the size of the array will constantly increase, and all functions may not fit into the gas block, since they have unlimited cycles.

Recommendation:
use the EnumerableSet from openzeppelin. There, all functions are performed as a constant and there is a possibility of iteration.

MEDIUM | UNRESOLVED

#54 line. The rate not multiplied by the amount, like in line #64.

LOW | UNRESOLVED

There is no point in onlyOwner modifier on getProviders getters since all data is in the open form in the blockchain and the user can still get it.

Deposit.sol

HIGH | UNRESOLVED

If in the init function msg.sender is not equal to owner, then there will be an error in line #50, because the sender of the transaction will not have the right to add an operator.

Recommendation:
When deploying contracts, you need to be very careful with this point.

MEDIUM | UNRESOLVED

There is a possibility of overflow when subtracting on line 113.

Recommendation:
Use safeMath or add require, which will prevent this case from happening.

LOW | UNRESOLVED

Strings library is not used.

LOW | UNRESOLVED

#60 line. Useless variable. To reduce the total bytecode of the contract, you can write in 1 line. For example, _deposits [swapId] [participant] = DepositLeg (...);

```
 99            _deposits[swapId][participant].assetType = assetType;
100            _deposits[swapId][participant].assetAddress = asset;
101            _deposits[swapId][participant].amount = amount;
102            _deposits[swapId][participant].time = lockedTime;
```

There are a lot of storage reads in the redeem function.

Recommendation:
Move the most frequently used variables into local variables.

LOW | UNRESOLVED

Line #60. We can use the _participant variable, since we have checked above that it is equivalent.

```
58            //todo: borrow from safebox
59            if (_participant == addressProvider.getSafeBoxAddress()) {
60                SafeBox safeBox = SafeBox(address(uint160(addressProvider.getSafeBoxAddress())));
61                safeBox.borrowERC20(_asset, _amount);
```

EquiToken.sol

CRITICAL | UNRESOLVED

The __initializeGrantor function of the GrantorRole contract is not called in the init function.

CRITICAL | UNRESOLVED

Contracts do not compile due to incorrect init function.

```
11    function initialize(string memory name, string memory symbol) public initializer {
12        // Ownable
13        super.initialize(_msgSender());
14        // ERC20PresetMinterPauserUpgradeable
15        super.overrideinitialize(name, symbol);
16        // ERC20CappedUpgradeable
17        __ERC20Capped_init(CAP);
18        // decimals
19        _setupDecimals(DECIMALS);
20    }
```

SafeBox.sol

MEDIUM | UNRESOLVED

Reason strings are needed in lines 37 and 54.

LOW | UNRESOLVED

It is possible to use an interface rather than a complete contract. This will save bytecode.

```
28    function initialize(address _owner, AddressProvider _addressProvider) initializer public {
29        super.initialize(_owner);
30        addressProvider = _addressProvider;
31    }
```

Recommendation:
Use call instead of a transfer for the Ether. Since the gasLimit for the transfer is limited to a constant number, sometimes it happens that there is not enough gas for the transfer. That is, the Ether network has updated this number, and contracts for the old version of solidity, etc.

Vestable.sol

MEDIUM | UNRESOLVED

Duplicate check on lines 169 and 227. If the grantor makes a mistake and passes the start date, which has not yet arrived, then an error will occur on line 298.

```
297            // Compute the exact number of days vested.
298            uint32 daysVested = onDay - grant.startDay;
```

LOW | UNRESOLVED

Instead of the SECONDS_PER_DAY constant, you can use the built-in function '1 days'.

LOW | UNRESOLVED

In line #57 (at the end of the line) 'ok' should be avoided if it is not used. Please check that in all functions.

```
56
57      function _hasVestingSchedule(address account) internal view returns (bool ok) {
58          return _vestingSchedules[account].isValid;
59      }
```

LOW | UNRESOLVED

Structures must start with a capital letter.

SwapProtocol.sol

HIGH | UNRESOLVED

Functions 1 and 2 in contract 3 have modifier 4. It was not evident from the code that the SwapProtocol contract was made somewhere by the operator. Because of this, the swapERC20 and redeemSwap functions will not work.

Recommendation:
Fix this point in the code of migrations.

MEDIUM | UNRESOLVED

Instead of _swapDivisor, it is better to use the same number everywhere. Usually, 10 ^ 27 is taken as 100%. This contract uses 10 ^ 18, so you can use this constant wherever there is division. Respectively move it into a separate variable.

MEDIUM | UNRESOLVED

In the getSwapById function, you can return a structure object. To do this, you need to connect ABIEncoderV2.

MEDIUM | UNRESOLVED

The swapType argument in many functions is not clear what it entails.

Recommendation:
Use enumeration with the required types.

LOW | UNRESOLVED

A similar thing has already been described above. Can be done in one time to reduce bytecode.

```
110         _swaps[newSwapID].swapType = swapType;
111         _swaps[newSwapID].openTime = now;
112         _swaps[newSwapID].duration = swapPeriod;
113         _swaps[newSwapID].assetType = _ERC20;
114         _swaps[newSwapID].participantAddress = msg.sender;
115         _swaps[newSwapID].participantCollateral = participantCollateral;
116         _swaps[newSwapID].ownerCollateral = ownerCollateral;
117         _swaps[newSwapID].cTokenAddress = cTokenAddress;
118         _swaps[newSwapID].cTokenAmount = cTokenAmount;
119         _swaps[newSwapID].uTokenAmount = uTokenAmount;
```

LOW | UNRESOLVED

The swapETH function is useless in this case.

LOW | UNRESOLVED

Use block.timestamp instead of ‘now’, because ‘now’ is deprecated.

LOW | UNRESOLVED

Remove Strings library.

General Tips

- In some places, there are unnecessary conversions to types.
  For example (In SwapProtocol.sol file):

```
207     uint256 _collateralRate = uint256(0);
```

```
134     require(address(_swaps[swapId].participantAddress)
```

- If the public function is not called anywhere from other contracts, then it is better to make it external, since they are cheaper in terms of gas.

CODE COVERAGE AND TEST RESULTS FOR ALL FILES

Tests written by Zokyo Secured team

As part of our work assisting EqiFi in verifying the correctness of their contract code, our team was responsible for writing integration tests using the Truffle testing framework.

Tests were based on the functionality of the code, as well as a review of the EqiFi contract requirements for details about issuance amounts and how the system handles these.

```
Contract: CurrencySwap
  ✓ setup (3568ms)
  on gets/sets
    ✓ safebox (657ms)
    ✓ deposit (602ms)
    ✓ currency swap (565ms)
    ✓ interest rate oracle (619ms)
    ✓ interest rate calculator (681ms)
    ✓ swap protocol (615ms)
  only admin
    ✓ should Not set safebox address (902ms)
    ✓ should Not set deposit address (342ms)
    ✓ should Not set currency swap address (319ms)
    ✓ should Not set interest rate oracle address (388ms)
    ✓ should Not set interest rate calculator address (318ms)
    ✓ should Not set swap protocol address address (297ms)

Contract: CurrencySwap
  ✓ setup (3114ms)
  ✓ should call getPrice (254ms)
  ✓ should call swap (291ms)

Contract: Deposit
  ✓ setup (4564ms)
  on eth
    ✓ should NOT deposit [sender is not operaton] (406ms)
```

```
    ✓ should NOT deposit [swapId = 0] (267ms)
    ✓ should NOT deposit [zero address] (545ms)
    ✓ should NOT deposit [amount == 0] (288ms)
    ✓ should NOT deposit [time lock == 0] (363ms)
    ✓ should deposit [user1] (1408ms)
    ✓ should NOT deposit [same id, same user] (301ms)
    ✓ should deposit [user2] (1365ms)
    ✓ should NOT deposit [id is full] (326ms)
  on ERC20
    ✓ should get tokens (1604ms)
    ✓ should NOT deposit [transfer not allowed] (684ms)
    ✓ should deposit (4063ms)
  on redeem
    on eth
      ✓ should NOT call redeem [1 deposit] (1831ms)
      ✓ should NOT call redeem [other account] (235ms)
      ✓ should call redeem [ETH] (451ms)
      ✓ should NOT call redeem [ETH, already redeemed] (269ms)
      ✓ should NOT call redeem [locked] (3408ms)
    on erc20
      ✓ should call redeem [ERC20] (1069ms)

Contract: InterestRateCalculator
  ✓ setup (2316ms)
  ✓ should call getLoanInterestRate (321ms)
  ✓ should call getDepositInterestRate (281ms)

Contract: Ownership
  ✓ setup (6678ms)
  ✓ should trasfer operator (1162ms)
  ✓ should NOT transfeer owner [new owner is 0x000...] (2000ms)
  ✓ should transfer ownership (1422ms)
  ✓ should NOT call func [not owner] (162ms)

Contract: SwapProtocol
  ✓ setup (2408ms)
  ✓ should call swap (362ms)
  ✓ should call getSwapInterestRate (333ms)
```

```
    ✓ should call redeemSwap (305ms)

Contract: SafeBox
    ✓ setup (2850ms)
    on receive
      ✓ balance should be 0
      ✓ should transfer 10 eth (302ms)
    on withdraw
      on SafeBox
        ✓ should withdraw 10 eth (387ms)
        ✓ should NOT withdrow 10000 eth (381ms)
      on ERC20
        ✓ should withdraw 10 eth (3907ms)
    on borrow
      on SafeBox
        ✓ should borrow from deposit address (1317ms)
        ✓ should NOT borrow [address is not deposit] (932ms)
      on ERC20
        ✓ borrowERC20 - deposit address should be able to borrow erc20 added by owner (2579ms)
        ✓ borrowERC20 - only deposit address should be able to borrow erc20 added by owner (17400ms)

  57 passing (1m)
```

FILE	% STMTS	% BRANCH	% FUNCS	% LINES	UNCOVERED LINES
contracts\	95.85	94.44	100.00	98.85	
AddressProvider.sol	100.00	100.00	100.00	100.00	
AddressStorage.sol	100.00	100.00	100.00	100.00	
CurrencySwap.sol	100.00	100.00	100.00	100.00	
Deposit.sol	97.50	96.43	100.00	97.44	101
InterestRateCalculator.sol	100.00	100.00	100.00	100.00	
SafeBox.sol	100.00	87.50	100.00	100.00	
SwapProtocol.sol	100.00	100.00	100.00	100.00	
contracts\functional	100.00	87.50	100.00	100.00	
Accessible.sol	100.00	75.00	100.00	100.00	
Ownable.sol	100.00	100.00	100.00	100.00	
All files	**99.04**	**93.18**	**100.00**	**99.06**	

We are grateful to have been given the opportunity to work with the EqiFi team.

The statements made in this document should not be interpreted as investment or legal advice, nor should its authors be held accountable for decisions made based on them.

Zokyo's Security Team recommends that the EqiFi team put in place a bug bounty program to encourage further analysis of the smart contract by third parties.

ZOKYO.