0.7

bZx v2 (0.7) Process Quality Review

Score: 76%

Overview

This is a bZx Process Quality Review completed on September 2nd 2021. It was performed using the Process Review process (version 0.7.3) and is documented here. The review was performed by Nic of DeFiSafety. Check out our Telegram.

The final score of the review is **76%**, a **PASS**. The breakdown of the scoring is in Scoring Appendix. For our purposes, a pass is **70%.**

Summary of the Process

Very simply, the review looks for the following declarations from the developer's site. With these declarations, it is reasonable to trust the smart contracts.

- **Here are my smart contracts on the blockchain**
- **Here is the documentation that explains what my smart contracts do**
- **Here are the tests I ran to verify my smart contract**
- **Here are the audit(s) performed on my code by third party experts**
- **Here are the admin controls and strategies**

Disclaimer

This report is for informational purposes only and does not constitute investment advice of any kind, nor does it constitute an offer to provide investment advisory or other services. Nothing in this report shall be considered a solicitation or offer to buy or sell any security, token, future, option or other financial instrument or to offer or provide any investment advice or service to any person in any jurisdiction. Nothing contained in this report constitutes investment advice or offers any opinion with respect to the suitability of any security, and the views expressed in this report should not be taken as advice to buy, sell or hold any security. The information in this report should not be relied upon for the purpose of investing. In preparing the information contained in this report, we have not taken into account the investment needs, objectives and financial circumstances of any particular investor. This information has no regard to the specific investment objectives, financial situation and particular needs of any specific recipient of this information and investments discussed may not be suitable for all investors.

Any views expressed in this report by us were prepared based upon the information available to us at the time such views were written. The views expressed within this report are limited to DeFiSafety and the author and do not reflect those of any additional or third party and are strictly based upon DeFiSafety, its authors, interpretations and evaluation of relevant data. Changed or additional information could cause such

views to change. All information is subject to possible correction. Information may quickly become unreliable for various reasons, including changes in market conditions or economic circumstances.

Chain

This section indicates the blockchain used by this protocol.

> **Chain:** Ethereum, Polygon, Binance Smart Chain

Guidance:

Ethereum
Binance Smart Chain
Polygon
Avalanche
Terra

Code and Team

This section looks at the code deployed on the Mainnet that gets reviewed and its corresponding software repository. The document explaining these questions is here. This review will answer the following questions:

1) Are the executing code addresses readily available? (%)
2) Is the code actively being used? (%)
3) Is there a public software repository? (Y/N)
4) Is there a development history visible? (%)
5) Is the team public (not anonymous)? (Y/N)

1) Are the executing code addresses readily available? (%)

> **Answer:** 40%

They are available at website https://github.com/bZxNetwork/bzx-subgraph/blob/main/subgraph.yaml, as indicated in the Appendix.

Guidance:

100% Clearly labelled and on website, docs or repo, quick to find
70% Clearly labelled and on website, docs or repo but takes a bit of looking

40%	Addresses in mainnet.json, in discord or sub graph, etc
20%	Address found but labeling not clear or easy to find
0%	Executing addresses could not be found

How to improve this score:

Make the Ethereum addresses of the smart contract utilized by your application available on either your website or your GitHub (in the README for instance). Ensure the addresses is up to date. This is a very important question towards the final score.

2) Is the code actively being used? (%)

> **Answer:** 100%

Activity is over 10 transactions a day on contract *ProxyStaking.sol*, as indicated in the Appendix.

Guidance:

100%	More than 10 transactions a day
70%	More than 10 transactions a week
40%	More than 10 transactions a month
10%	Less than 10 transactions a month
0%	No activity

3) Is there a public software repository? (Y/N)

> **Answer:** Yes

GitHub: https://github.com/bZxNetwork/contractsV2.

Is there a public software repository with the code at a minimum, but also normally test and scripts. Even if the repository was created just to hold the files and has just 1 transaction, it gets a **"Yes"**. For teams with private repositories, this answer is **"No"**.

4) Is there a development history visible? (%)

> **Answer:** 100%

With 859 commits and 78 branches, the bZx v2 contracts repository is very healthy.

This metric checks if the software repository demonstrates a strong steady history. This is normally demonstrated by commits, branches and releases in a software repository. A healthy history demonstrates a history of more than a month (at a minimum).

Guidance:

100% Any one of 100+ commits, 10+branches
70% Any one of 70+ commits, 7+branches
50% Any one of 50+ commits, 5+branches
30% Any one of 30+ commits, 3+branches
0% Less than 2 branches or less than 30 commits

5) Is the team public (not anonymous)? (Y/N)

> **Answer:** Yes

Location: https://angel.co/company/bzx-1/people.

For a **"Yes"** in this question, the real names of some team members must be public on the website or other documentation (LinkedIn, etc). If the team is anonymous, then this question is a **"No"**.

Documentation

This section looks at the software documentation. The document explaining these questions is here.

Required questions are;

6) Is there a whitepaper? (Y/N)
7) Are the basic software functions documented? (Y/N)
8) Does the software function documentation fully (100%) cover the deployed contracts? (%)
9) Are there sufficiently detailed comments for all functions within the deployed contract code (%)
10) Is it possible to trace from software documentation to the implementation in code (%)

6) Is there a whitepaper? (Y/N)

> **Answer:** Yes

Location: https://docs.bzx.network/.

7) Are the basic software functions documented? (Y/N)

> **Answer:** Yes

There are basic software functions documented in the "Integration" section of the bZx documentation.

8) Does the software function documentation fully (100%) cover the deployed contracts? (%)

> **Answer:** 80%

There major basic software functions are documented in the "Integration" section of the bZx documentation.

Guidance:

100% All contracts and functions documented
80% Only the major functions documented
79-1% Estimate of the level of software documentation
0% No software documentation

How to improve this score:

This score can be improved by adding content to the software functions document such that it comprehensively covers the requirements. For guidance, refer to the SecurEth System Description Document. Using tools that aid traceability detection will help.

9) Are there sufficiently detailed comments for all functions within the deployed contract code (%)

> **Answer:** 0%

Code examples are in the Appendix. As per the SLOC, there is 11% commenting to code (CtC).

The Comments to Code (CtC) ratio is the primary metric for this score.

Note: No testnet, interface, or third-party files were used in the calculation of the CtC.

Guidance:

100% CtC > 100 Useful comments consistently on all code
90-70% CtC > 70 Useful comment on most code
60-20% CtC > 20 Some useful commenting
0% CtC < 20 No useful commenting

How to improve this score

This score can improve by adding comments to the deployed code such that it comprehensively covers the code. For guidance, refer to the SecurEth Software Requirements.

10) Is it possible to trace from software documentation to the implementation in code (%)

> **Answer:** 60%

There is a clear association between the software documentation and the code, but there is no explicit traceability as to its implementation in the bZx source code.

Guidance:

100% Clear explicit traceability between code and documentation at a requirement level for all code
60% Clear association between code and documents via non explicit traceability
40% Documentation lists all the functions and describes their functions
0% No connection between documentation and code

How to improve this score:

This score can improve by adding traceability from documentation to code such that it is clear where each outlined function is coded in the source code. For reference, check the SecurEth guidelines on traceability.

Testing

This section looks at the software testing available. It is explained in this document. This section answers the following questions;

11) Full test suite (Covers all the deployed code) (%)
12) Code coverage (Covers all the deployed lines of code, or explains misses) (%)
13) Scripts and instructions to run the tests (Y/N)
14) Report of the results (%)
15) Formal Verification test done (%)
16) Stress Testing environment (%)

11) Is there a Full test suite? (%)

> **Answer:** 40%

Code examples are in the Appendix. As per the SLOC, there is 47% testing to code (TtC).

This score is guided by the Test to Code ratio (TtC). Generally a good test to code ratio is over 100%. However the reviewers best judgement is the final deciding factor.

Guidance:

100% TtC > 120% Both unit and system test visible
80% TtC > 80% Both unit and system test visible
40% TtC < 80% Some tests visible
0% No tests obvious

How to improve this score:

This score can improved by adding tests to fully cover the code. Document what is covered by traceability or test results in the software repository.

12) Code coverage (Covers all the deployed lines of code, or explains misses) (%)

> **Answer:** 30%

No evidence of code coverage in the bZx documentation or in their audit reports. However, some tests are evident, but also incomplete.

Guidance:

100%	Documented full coverage
99-51%	Value of test coverage from documented results
50%	No indication of code coverage but clearly there is a reasonably complete set of tests
30%	Some tests evident but not complete
0%	No test for coverage seen

How to improve this score:

This score can improved by adding tests that achieve full code coverage. A clear report and scripts in the software repository will guarantee a high score.

13) Scripts and instructions to run the tests (Y/N)

> **Answer:** Yes

Scrips/Instructions location: https://github.com/bZxNetwork/contractsV2/tree/development/scripts.

14) Report of the results (%)

> **Answer:** 0%

No test reports were found in any of the bZx documentation or GitHub repositories.

Guidance:

100%	Detailed test report as described below
70%	GitHub code coverage report visible
0%	No test report evident

How to improve this score

Add a report with the results. The test scripts should generate the report or elements of it.

15) Formal Verification test done (%)

> **Answer:** 100%

A bZx Formal Verification test was performed by Certik.

16) Stress Testing environment (%)

> **Answer:** 100%

There is evidence of bZx smart contract stress testing at https://github.com/bZxNetwork/bzx-subgraph/blob/main/config/mainnet-test.json.

Security

This section looks at the 3rd party software audits done. It is explained in this document. This section answers the following questions;

17) Did 3rd Party audits take place? (%)
18) Is the bounty value acceptably high?

17) Did 3rd Party audits take place? (%)

> **Answer:** 100%

Multiple audits were performed before and after each of the bZx v1 and v2 launches. A full list of all the audits can be found at https://bzx.network/security.

Guidance:

100% Multiple Audits performed before deployment and results public and implemented or not required
90% Single audit performed before deployment and results public and implemented or not required
70% Audit(s) performed after deployment and no changes required. Audit report is public

50% Audit(s) performed after deployment and changes needed but not implemented
20% No audit performed
0% Audit Performed after deployment, existence is public, report is not public and no improvements deployed OR smart contract address' not found, (where question 1 is 0%)

Deduct 25% if code is in a private repo and no note from auditors that audit is applicable to deployed code

18) Is the bounty value acceptably high (%)

> ⓘ **Answer:** 70%

bZx's Bug Bounty program rewards participating users with up to 350k for the most critical of finds.

Guidance:

100% Bounty is 10% TVL or at least $1M AND active program (see below)
90% Bounty is 5% TVL or at least 500k AND active program
80% Bounty is 5% TVL or at least 500k
70% Bounty is 100k or over AND active program
60% Bounty is 100k or over
50% Bounty is 50k or over AND active program
40% Bounty is 50k or over
20% Bug bounty program bounty is less than 50k
0% No bug bounty program offered

An active program means that a third party (such as Immunefi) is actively driving hackers to the site. An inactive program would be static mentions on the docs.

Access Controls

This section covers the documentation of special access controls for a DeFi protocol. The admin access controls are the contracts that allow updating contracts or coefficients in the protocol. Since these contracts can allow the protocol admins to "change the rules", complete disclosure of capabilities is vital for user's transparency. It is explained in this document. The questions this section asks are as follow;

19) Can a user clearly and quickly find the status of the admin controls?
20) Is the information clear and complete?
21) Is the information in non-technical terms that pertain to the investments?
22) Is there Pause Control documentation including records of tests?

19) Can a user clearly and quickly find the status of the access controls (%)

> ⓘ **Answer:** 70%

bZx's access control documentation was found after looking through their blog section on their website. Article introducing the bZx DAO.

Guidance:

100% Clearly labelled and on website, docs or repo, quick to find
70% Clearly labelled and on website, docs or repo but takes a bit of looking
40% Access control docs in multiple places and not well labelled
20% Access control docs in multiple places and not labelled
0% Admin Control information could not be found

20) Is the information clear and complete (%)

> **Answer:** 90%

a) Contracts are clearly labelled as upgradeable through the bZx voting architecture.

b) There are clear and defined Judicial, Executive, and Legislative roles that are distributed to members of the DAO.

c) The capabilities for change in the contract, and what the users can vote on, are described.

Note: The staking contract's code can be upgraded through the implementation of a "*StakingUpgradeable.sol*" contract. The swap contract's code can be upgraded through the use of calls to interface contracts, as well as the *delegatecall* function. The governance contract's code can also be upgraded through the use of the *Initialize* function. Lastly, the farm contracts, specifically the MasterChef ones, all have a "*Upgradeable.sol*" contract called.

Guidance:

All the contracts are immutable -- 100% OR

a) All contracts are clearly labelled as upgradeable (or not) -- 30% AND
b) The type of ownership is clearly indicated (OnlyOwner / MultiSig / Defined Roles) -- 30% AND
c) The capabilities for change in the contracts are described -- 30%

How to improve this score:

Create a document that covers the items described above. An example is enclosed.

21) Is the information in non-technical terms that pertain to the investments (%)

> **Answer:** 90%

bZx uses user-friendly language in order to inform their DAO community as to how their protocol and funds are in safe hands at https://bzx.network/blog/introducing-bzxdao.

Note: The staking contract's code can be upgraded through the implementation of a "*StakingUpgradeable.sol*" contract. The swap contract's code can be upgraded through the use of calls to interface contracts, as well as the *delegatecall* function. The governance contract's code can also be upgraded through the use of the *Initialize* function. Lastly, the farm contracts, specifically the MasterChef ones, all have a "*Upgradeable.sol*" contract called.

Guidance:

100%	All the contracts are immutable
90%	Description relates to investments safety and updates in clear, complete non-software l language
30%	Description all in software specific language
0%	No admin control information could not be found

How to improve this score:

Create a document that covers the items described above in plain language that investors can understand. An example is enclosed.

22) Is there Pause Control documentation including records of tests (%)

> **Answer:** 40%

bZx does not mention any Pause Control or a similar function in any of their documentation. However, there is a *PausableGuardian* contract in their software repository that acts as a Pause Control.

Guidance:

100%	All the contracts are immutable or no pause control needed and this is explained OR
100%	Pause control(s) are clearly documented and there is records of at least one test within 3 months
80%	Pause control(s) explained clearly but no evidence of regular tests
40%	Pause controls mentioned with no detail on capability or tests
0%	Pause control not documented or explained

How to improve this score**:**

Create a document that covers the items described above in plain language that investors can understand. An example is enclosed.

Appendices

Author Details

The author of this review is Rex of DeFi Safety.

Email : rex@defisafety.com Twitter : @defisafety

I started with Ethereum just before the DAO and that was a wonderful education. It showed the importance of code quality. The second Parity hack also showed the importance of good process. Here my aviation background offers some value. Aerospace knows how to make reliable code using quality processes.

I was coaxed to go to EthDenver 2018 and there I started SecuEth.org with Bryant and Roman. We created guidelines on good processes for blockchain code development. We got EthFoundation funding to assist in their development.

Process Quality Reviews are an extension of the SecurEth guidelines that will further increase the quality processes in Solidity and Vyper development.

DeFiSafety is my full time gig and we are working on funding vehicles for a permanent staff.

Scoring Appendix

	Total	bZx v2 0.7 UPDATE	
PQ Audit Scoring Matrix (v0.7)	**Points**	**Answer**	**Points**
Total	**260**		198
Code and Team			**76%**
1) Are the executing code addresses readily available? (%)	**20**	40%	8
2) Is the code actively being used? (%)	**5**	100%	5
3) Is there a public software repository? (Y/N)	**5**	Y	5
4) Is there a development history visible? (%)	**5**	100%	5
5) Is the team public (not anonymous)? (Y/N)	**15**	Y	15
Code Documentation			
6) Is there a whitepaper? (Y/N)	**5**	Y	5
7) Are the basic software functions documented? (Y/N)	**10**	Y	10
8) Does the software function documentation fully (100%) cov	**15**	80%	12
9) Are there sufficiently detailed comments for all functions w	**5**	0%	0
10) Is it possible to trace from software documentation to the	**10**	60%	6
Testing			
11) Full test suite (Covers all the deployed code) (%)	**20**	40%	8
12) Code coverage (Covers all the deployed lines of code, or ex	**5**	30%	1.5
13) Scripts and instructions to run the tests? (Y/N)	**5**	y	5
14) Report of the results (%)	**10**	0%	0
15) Formal Verification test done (%)	**5**	100%	5
16) Stress Testing environment (%)	**5**	100%	5
Security			
17) Did 3rd Party audits take place? (%)	**70**	100%	70
18) Is the bug bounty acceptable high? (%)	**10**	70%	7
Access Controls			
19) Can a user clearly and quickly find the status of the admin	**5**	70%	3.5
20) Is the information clear and complete	**10**	90%	9
21) Is the information in non-technical terms	**10**	90%	9
22) Is there Pause Control documentation including records of	**10**	40%	4
Section Scoring			
Code and Team	50	76%	
Documentation	45	73%	

Testing	50	49%
Security	80	96%
Access Controls	35	73%

Executing Code Appendix

```
specVersion: 0.0.2
description: bZx is a protocol for borrowing, lending, and margin trading
repository: https://github.com/bZxNetwork/bzx-subgraph
schema:
  file: ./schema.graphql
dataSources:
  - kind: ethereum/contract
    name: LoanOpeningsEvents
    network: mainnet
    source:
      address: '0xD8Ee69652E4e4838f2531732a46d1f7F584F0b7f'
      abi: LoanOpeningsEvents
      startBlock: 10600000
    mapping:
      kind: ethereum/events
      apiVersion: 0.0.4
      language: wasm/assemblyscript
      file: ./src/mappings/LoanOpeningsEvents.ts
      abis:
        - name: LoanOpeningsEvents
          file: ./abis/LoanOpeningsEvents.json
      entities:
          - BorrowEvent
          - TradeEvent
```

Code Used Appendix

0xc53980cb7ca5343ace...	13146888	42 mins ago	0xe95ebce2b02ee07def...	0x629b2ecf29dad224b2...	0 Ether
0xc53980cb7ca5343ace...	13146888	42 mins ago	0x9da41f7810c2548572f...	0xe95ebce2b02ee07def...	0 Ether
0x628c769a40071f9f837...	13145930	4 hrs 13 mins ago	0xe95ebce2b02ee07def...	0x6c3f90f043a72fa612c...	0 Ether
0x628c769a40071f9f837...	13145930	4 hrs 13 mins ago	0xe95ebce2b02ee07def...	0x629b2ecf29dad224b2...	0 Ether
0xb8b109149989ceb802...	13145271	6 hrs 43 mins ago	0xe95ebce2b02ee07def...	0x6c3f90f043a72fa612c...	0 Ether
0xb8b109149989ceb802...	13145271	6 hrs 43 mins ago	0xe95ebce2b02ee07def...	0x56d811088235f11c89...	0 Ether
0xb8b109149989ceb802...	13145271	6 hrs 43 mins ago	0xe95ebce2b02ee07def...	0x56d811088235f11c89...	0 Ether
0xb8b109149989ceb802...	13145271	6 hrs 43 mins ago	0xe95ebce2b02ee07def...	0x629b2ecf29dad224b2...	0 Ether
0x2d6367707de2af5889...	13145271	6 hrs 43 mins ago	0xe95ebce2b02ee07def...	0x6c3f90f043a72fa612c...	0 Ether
0x2d6367707de2af5889...	13145271	6 hrs 43 mins ago	0xe95ebce2b02ee07def...	0x56d811088235f11c89...	0 Ether
0x2d6367707de2af5889...	13145271	6 hrs 43 mins ago	0xe95ebce2b02ee07def...	0x56d811088235f11c89...	0 Ether
0x2d6367707de2af5889...	13145271	6 hrs 43 mins ago	0xe95ebce2b02ee07def...	0x629b2ecf29dad224b2...	0 Ether
0xf5ba31a5427b01a59a...	13140596	23 hrs 54 mins ago	0xe95ebce2b02ee07def...	0x6c3f90f043a72fa612c...	0 Ether
0xf5ba31a5427b01a59a...	13140596	23 hrs 54 mins ago	0xe95ebce2b02ee07def...	0x56d811088235f11c89...	0 Ether
0xf5ba31a5427b01a59a...	13140596	23 hrs 54 mins ago	0xe95ebce2b02ee07def...	0x56d811088235f11c89...	0 Ether
0xf5ba31a5427b01a59a...	13140596	23 hrs 54 mins ago	0xe95ebce2b02ee07def...	0x629b2ecf29dad224b2...	0 Ether

Example Code Appendix

```
contract StakingV1_1 is StakingState, StakingConstants, PausableGuardian {
    using MathUtil for uint256;

    modifier onlyEOA() {
        require(msg.sender == tx.origin, "unauthorized");
        _;
    }

    function getCurrentFeeTokens()
        external
        view
        returns (address[] memory)
    {
        return currentFeeTokens;
    }

    function _pendingSushiRewards(address _user)
        internal
        view
        returns (uint256)
    {
        uint256 pendingSushi = IMasterChefSushi(SUSHI_MASTERCHEF)
            .pendingSushi(BZRX_ETH_SUSHI_MASTERCHEF_PID, address(this));

        return _pendingAltRewards(
            SUSHI,
            _user,
            balanceOfByAsset(LPToken, _user),
            pendingSushi.mul(1e12).div(_totalSupplyPerToken[LPToken])
        );
    }

    function _pendingCrvRewards(address _user)
        internal
        returns (uint256)
    {
        uint256 pendingCrv = curve3PoolGauge.claimable_tokens(address(this));
        (,uint256 earnedStable,,) = _earned(_user, bzrxPerTokenStored, stableCoinPerTokenS
        uint256 res = _pendingAltRewards(
            CRV,
            _user,
            earnedStable,
            pendingCrv.mul(1e12).div(curve3PoolGauge.balanceOf(address(this)))
        );
        return res;
    }

    function _pendingAltRewards(address token, address _user, uint256 userSupply, uint256 
        internal

```

```
        view
        returns (uint256)
    {
        uint256 _altRewardsPerShare = altRewardsPerShare[token].add(extraRewardsPerShare);
        if (_altRewardsPerShare == 0)
            return 0;

        IStaking.AltRewardsUserInfo memory altRewardsUserInfo = userAltRewardsPerShare[_us
        return altRewardsUserInfo.pendingRewards.add(
                (_altRewardsPerShare.sub(altRewardsUserInfo.rewardsPerShare)).mul(userSupp
            );
    }

    function _depositToSushiMasterchef(uint256 amount)
        internal
    {
        uint256 sushiBalanceBefore = IERC20(SUSHI).balanceOf(address(this));
        IMasterChefSushi(SUSHI_MASTERCHEF).deposit(
            BZRX_ETH_SUSHI_MASTERCHEF_PID,
            amount
        );
        uint256 sushiRewards = IERC20(SUSHI).balanceOf(address(this)) - sushiBalanceBefore
        if (sushiRewards != 0) {
            _addAltRewards(SUSHI, sushiRewards);
        }
    }

    function _withdrawFromSushiMasterchef(uint256 amount)
        internal
    {
        uint256 sushiBalanceBefore = IERC20(SUSHI).balanceOf(address(this));
        IMasterChefSushi(SUSHI_MASTERCHEF).withdraw(
            BZRX_ETH_SUSHI_MASTERCHEF_PID,
            amount
        );
        uint256 sushiRewards = IERC20(SUSHI).balanceOf(address(this)) - sushiBalanceBefore
        if (sushiRewards != 0) {
            _addAltRewards(SUSHI, sushiRewards);
        }
    }


    function _depositTo3Pool(uint256 amount)
        internal
    {

        if(amount != 0)
            curve3PoolGauge.deposit(amount);

        //Trigger claim rewards from curve pool
        uint256 crvBalanceBefore = IERC20(CRV).balanceOf(address(this));
        curveMinter.mint(address(curve3PoolGauge));
        uint256 crvBalanceAfter = IERC20(CRV).balanceOf(address(this)) - crvBalanceBefore;
```

```
103
        if(crvBalanceAfter != 0){
104         _addAltRewards(CRV, crvBalanceAfter);
105         }
106     }
107
108     function _withdrawFrom3Pool(uint256 amount)
109         internal
110     {
111         if(amount != 0)
112             curve3PoolGauge.withdraw(amount);
113
114         //Trigger claim rewards from curve pool
115         uint256 crvBalanceBefore = IERC20(CRV).balanceOf(address(this));
116         curveMinter.mint(address(curve3PoolGauge));
117         uint256 crvBalanceAfter = IERC20(CRV).balanceOf(address(this)) - crvBalanceBefore;
118         if(crvBalanceAfter != 0){
119             _addAltRewards(CRV, crvBalanceAfter);
120         }
121     }
122
123
124     function stake(
125         address[] memory tokens,
126         uint256[] memory values
127     )
128         public
129         pausable
130         updateRewards(msg.sender)
131     {
132         require(tokens.length == values.length, "count mismatch");
133
134         /*address currentDelegate = delegate[msg.sender];
135         if (currentDelegate == address(0)) {
136             currentDelegate = msg.sender;
137             delegate[msg.sender] = currentDelegate;
138             _delegatedSet.addAddress(msg.sender);
139         }*/
140
141         address token;
142         uint256 stakeAmount;
143
144
145         for (uint256 i = 0; i < tokens.length; i++) {
146             token = tokens[i];
147             require(token == BZRX || token == vBZRX || token == iBZRX || token == LPToken,
148
149             stakeAmount = values[i];
150             if (stakeAmount == 0) {
151                 continue;
152             }
153             uint256 pendingBefore = (token == LPToken) ? _pendingSushiRewards(msg.sender)
154             _balancesPerToken[token][msg.sender] = _balancesPerToken[token][msg.sender].ad
155             _totalSupplyPerToken[token] = _totalSupplyPerToken[token].add(stakeAmount);
```

```
155             _totalSupplyPerToken[token] = _totalSupplyPerToken[token].add(stakeAmount);
156
157             /*delegatedPerToken[currentDelegate][token] = delegatedPerToken[currentDelegate
158                 .add(stakeAmount);*/
159
160             IERC20(token).safeTransferFrom(msg.sender, address(this), stakeAmount);
161
162             // Deposit to sushi masterchef
163             if (token == LPToken) {
164                 _depositToSushiMasterchef(
165                     IERC20(LPToken).balanceOf(address(this))
166                 );
167
168                 userAltRewardsPerShare[msg.sender][SUSHI] = IStaking.AltRewardsUserInfo({
169                         rewardsPerShare: altRewardsPerShare[SUSHI],
170                         pendingRewards: pendingBefore
171                     }
172                 );
173             }
174             emit Stake(
175                 msg.sender,
176                 token,
177                 msg.sender, //currentDelegate,
178                 stakeAmount
179             );
180         }
181     }
182
183     function unstake(
184         address[] memory tokens,
185         uint256[] memory values
186     )
187         public
188         pausable
189         updateRewards(msg.sender)
190     {
191         require(tokens.length == values.length, "count mismatch");
192
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complex
Solidity	83	18182	2545	1487	14150	1918

Comments to Code 1487/14150 = 11%

Javascript Tests

Language	Files	Lines	Blanks	Comments	Code	Complex
JavaScript	68	8418	1408	437	6573	81

Tests to Code 6573/14150 = 47%