0.7

Polycat Finance Process Quality Review

Score: 38%

Overview

This is a Polycat Process Quality Review completed on July 12th 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 38%, a fail. 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: Polygon**

Guidance:

Ethereum
Binance Smart Chain
Polygon

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: 100%

They are available at website https://docs.polycat.finance/contracts, 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

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

> ✓ Answer: 100%

Activity is 80,000 transactions a day on contract *MasterChef.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/polycatfi

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: 0%

With 10 commits and 1 branch, this is an unhealthy software repository.

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

How to improve this score:

Continue to test and perform other verification activities after deployment, including routine maintenance updating to new releases of testing and deployment tools. A public development history indicates clearly to the public the level of continued investment and activity by the developers on the application. This gives a level of security and faith in the application.

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

> ⚠ Answer: No

Team is made up of two anonymous developers as mentioned here.

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.polycat.finance/.

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

> ⚠ Answer: No

There are no basic software functions (code) listed or documented in their documentation.

How to improve this score:

Write the document based on the deployed code. For guidance, refer to the SecurEth System Description Document.

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

> ⚠ Answer: 0%

There are no software functions (code) listed or documented in their 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 14% commenting to code (CtC).

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

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: 0%

As there are no software functions (code) detailed in their documentation, there isn't any traceability as to their implementation in code either.

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: 0%

There is no testing suite inside of Polycat's public GitHub repository.

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: 0%

There is no evidence of Polycat code coverage in any of their software repositories or in their audits by Obelisk and TechRate.

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: No

As there is no testing suite in Polycat's GitHub repository, there are not any scripts or test instructions either.

How to improve this score:

Add the scripts to the repository and ensure they work. Ask an outsider to create the environment and run the tests. Improve the scripts and docs based on their feedback.

14) Report of the results (%)

> ⚠ Answer: 0%

There is no evidence of any test results inside of Polycat's software 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: 0%

There isn't any evidence of a Polycat Formal Verification test in any of their documentation or in web searches.

16) Stress Testing environment (%)

⚠ Answer: 0%

Although the team mentions their long usage of BSC test-net at https://polycatfinance.medium.com/polycat-finance-our-first-adventure-on-polygon-86ecc876744a, there are no published testnet addresses to verify this information.

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: 90%

Obelisk published their first Polycat audit report on May 19th 2021.

Obelisk published their second Polycat audit report on June 2nd 2021.

TechRate published a Polycat audit report in early May 2021.

Polycat was launched on May 2nd 2021.

Note: Most fix recommendations by the audits were implemented successfully by the Polycat team.

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, question

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: 0%

No evidence of a Polycat Bug Bounty program was found in their documentation or in web searches.

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: 0%

There is no admin access control information in any of the Polycat documentation.

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: 0%

There are no specific details regarding contract upgradeability, ownership type, or capabilities for change in contracts described in the Polycat documentation.

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: 0%

There is no admin access control information in any of the Polycat documentation.

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: 0%

There is no evidence of Pause Control or similar function documented in the Polycat documentation or GitHub repository.

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	PolyCat Finance	
PQ Audit Scoring Matrix (v0.7)	**Points**	**Answer**	**Points**
Total	**260**		98
Code and Team			**38%**
3) Is there a public software repository? (Y/N)	**5**	Y	5
4) Is there a development history visible? (%)	**5**	0%	0
5) Is the team public (not anonymous)? (Y/N)	**15**	N	0
Code Documentation			
6) Is there a whitepaper? (Y/N)	**5**	Y	5
7) Are the basic software functions documented? (Y/N)	**10**	N	0
8) Does the software function documentation fully (100%) cover the deployed contracts? (%)	**15**	0%	0
9) Are there sufficiently detailed comments for all functions within the deployed contract code (%)	**5**	0%	0
10) Is it possible to trace from software documentation to the implementation in code (%)	**10**	0%	0
Testing			
11) Full test suite (Covers all the deployed code) (%)	**20**	0%	0
12) Code coverage (Covers all the deployed lines of code, or explains misses) (%)	**5**	0%	0
13) Scripts and instructions to run the tests? (Y/N)	**5**	0	0
14) Report of the results (%)	**10**	0%	0
15) Formal Verification test done (%)	**5**	0%	0
16) Stress Testing environment (%)	**5**	0%	0
Security			
17) Did 3rd Party audits take place? (%)	**70**	90%	63
18) Is the bug bounty acceptable high? (%)	**10**	0%	0
Access Controls			
19) Can a user clearly and quickly find the status of the admin controls	**5**	0%	0
20) Is the information clear and complete	**10**	0%	0
21) Is the information in non-technical terms	**10**	0%	0
22) Is there Pause Control documentation including records of tests	**10**	0%	0
Section Scoring			
Code and Team	50	60%	
Documentation	45	11%	
Testing	50	0%	
Security	80	79%	
Access Controls	35	0%	

Executing Code Appendix

Chef Contracts

MasterChef `0x8CFD1B9B7478E7B0422916B72d1DB6A9D513D734`

- The **MasterChef** controls the Polycat Farms and Pools functions.

VaultChef `0xBdA1f897E851c7EF22CD490D2Cf2DAce4645A904`

- The **VaultChef** controls the Polycat Vault functions.

Other Contracts

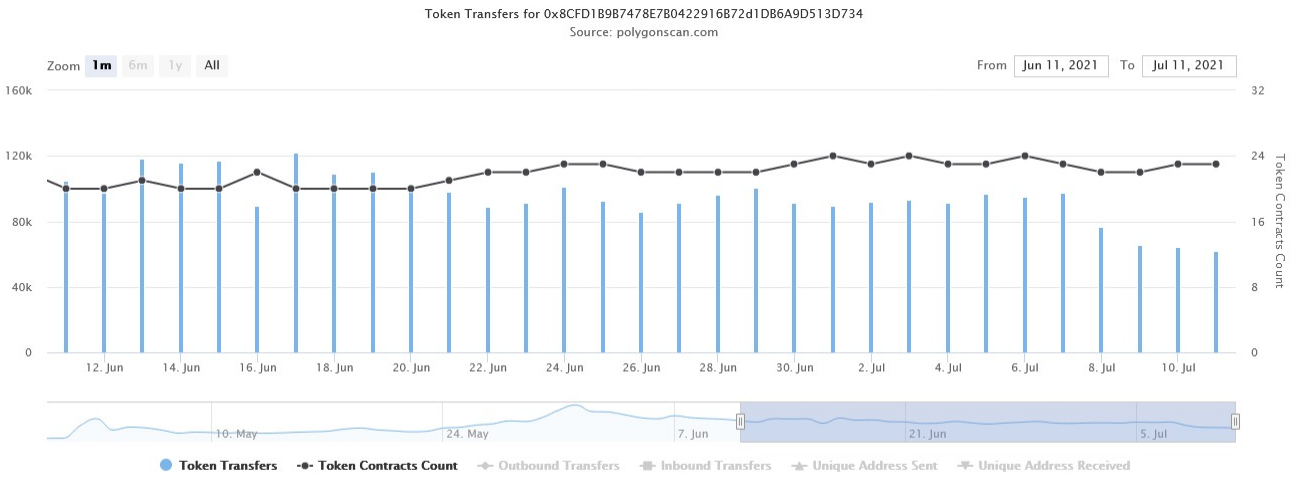
Timelock `0xf5a824B077Cc0aaF50Cf83a9E82714b89B684925`

- The **Timelock** prevents instant changes to our smart contracts.

Referral Handler `0xB67aD6C2Fe7dd6Ba346706b833cCF4234256266D`

- The **Referral Handler** tracks and rewards referrals.

Code Used Appendix



Pro-Tip: Click on the chart data points to view more

Example Code Appendix

```
// MasterChef is the master of Fish. He can make Fish and he is a fair guy.
//
// Note that it's ownable and the owner wields tremendous power. The ownership
// will be transferred to a governance smart contract once Fish is sufficiently
// distributed and the community can show to govern itself.
//
// Have fun reading it. Hopefully it's bug-free. God bless.
contract MasterChef is Ownable, ReentrancyGuard {
    using SafeMath for uint256;
    using SafeERC20 for IERC20;

    // Info of each user.
    struct UserInfo {
        uint256 amount;         // How many LP tokens the user has provided.
        uint256 rewardDebt;     // Reward debt. See explanation below.
        //
        // We do some fancy math here. Basically, any point in time, the amount of FISHes
        // entitled to a user but is pending to be distributed is:
        //
        //   pending reward = (user.amount * pool.accFishPerShare) - user.rewardDebt
        //
        // Whenever a user deposits or withdraws LP tokens to a pool. Here's what happens:
        //   1. The pool's `accFishPerShare` (and `lastRewardBlock`) gets updated.
        //   2. User receives the pending reward sent to his/her address.
        //   3. User's `amount` gets updated.
        //   4. User's `rewardDebt` gets updated.
    }

    // Info of each pool.
    struct PoolInfo {
        IERC20 lpToken;           // Address of LP token contract.
        uint256 allocPoint;       // How many allocation points assigned to this pool. FISH
        uint256 lastRewardBlock;  // Last block number that FISHes distribution occurs.
        uint256 accFishPerShare;   // Accumulated FISHes per share, times 1e18. See below.
        uint16 depositFeeBP;      // Deposit fee in basis points
    }

    // The FISH TOKEN!
    FishToken public fish;
    address public devAddress;
    address public feeAddress;
    address public vaultAddress;

    // FISH tokens created per block.
    uint256 public fishPerBlock = 1 ether;

    // Info of each pool.
    PoolInfo[] public poolInfo;
    // Info of each user that stakes LP tokens.

```

```
    mapping(uint256 => mapping(address => UserInfo)) public userInfo;
    // Total allocation points. Must be the sum of all allocation points in all pools.
    uint256 public totalAllocPoint = 0;
    // The block number when FISH mining starts.
    uint256 public startBlock;

    // Fish referral contract address.
    IReferral public referral;
    // Referral commission rate in basis points.
    uint16 public referralCommissionRate = 200;
    // Max referral commission rate: 5%.
    uint16 public constant MAXIMUM_REFERRAL_COMMISSION_RATE = 500;

    event Deposit(address indexed user, uint256 indexed pid, uint256 amount);
    event Withdraw(address indexed user, uint256 indexed pid, uint256 amount);
    event EmergencyWithdraw(address indexed user, uint256 indexed pid, uint256 amount);
    event SetFeeAddress(address indexed user, address indexed newAddress);
    event SetDevAddress(address indexed user, address indexed newAddress);
    event SetVaultAddress(address indexed user, address indexed newAddress);
    event SetReferralAddress(address indexed user, IReferral indexed newAddress);
    event UpdateEmissionRate(address indexed user, uint256 fishPerBlock);
    event ReferralCommissionPaid(address indexed user, address indexed referrer, uint256 co

    constructor(
        FishToken _fish,
        uint256 _startBlock,
        address _devAddress,
        address _feeAddress,
        address _vaultAddress
    ) public {
        fish = _fish;
        startBlock = _startBlock;

        devAddress = _devAddress;
        feeAddress = _feeAddress;
        vaultAddress = _vaultAddress;
    }

    function poolLength() external view returns (uint256) {
        return poolInfo.length;
    }

    mapping(IERC20 => bool) public poolExistence;
    modifier nonDuplicated(IERC20 _lpToken) {
        require(poolExistence[_lpToken] == false, "nonDuplicated: duplicated");
        _;
    }

    // Add a new lp to the pool. Can only be called by the owner.
    function add(uint256 _allocPoint, IERC20 _lpToken, uint16 _depositFeeBP) external onlyO
        require(_depositFeeBP <= 10000, "add: invalid deposit fee basis points");
        uint256 lastRewardBlock = block.number > startBlock ? block.number : startBlock;
        totalAllocPoint = totalAllocPoint.add(_allocPoint);
```

```
103
        poolExistence[_lpToken] = true;
104         poolInfo.push(PoolInfo({
105             lpToken: _lpToken,
106             allocPoint: _allocPoint,
107             lastRewardBlock: lastRewardBlock,
108             accFishPerShare: 0,
109             depositFeeBP: _depositFeeBP
110         }));
111     }
112
113     // Update the given pool's FISH allocation point and deposit fee. Can only be called by
114     function set(uint256 _pid, uint256 _allocPoint, uint16 _depositFeeBP) external onlyOwne
115         require(_depositFeeBP <= 10000, "set: invalid deposit fee basis points");
116         totalAllocPoint = totalAllocPoint.sub(poolInfo[_pid].allocPoint).add(_allocPoint);
117         poolInfo[_pid].allocPoint = _allocPoint;
118         poolInfo[_pid].depositFeeBP = _depositFeeBP;
119     }
120
121     // Return reward multiplier over the given _from to _to block.
122     function getMultiplier(uint256 _from, uint256 _to) public pure returns (uint256) {
123         return _to.sub(_from);
124     }
125
126     // View function to see pending FISHes on frontend.
127     function pendingFish(uint256 _pid, address _user) external view returns (uint256) {
128         PoolInfo storage pool = poolInfo[_pid];
129         UserInfo storage user = userInfo[_pid][_user];
130         uint256 accFishPerShare = pool.accFishPerShare;
131         uint256 lpSupply = pool.lpToken.balanceOf(address(this));
132         if (block.number > pool.lastRewardBlock && lpSupply != 0) {
133             uint256 multiplier = getMultiplier(pool.lastRewardBlock, block.number);
134             uint256 fishReward = multiplier.mul(fishPerBlock).mul(pool.allocPoint).div(tota
135             accFishPerShare = accFishPerShare.add(fishReward.mul(1e18).div(lpSupply));
136         }
137         return user.amount.mul(accFishPerShare).div(1e18).sub(user.rewardDebt);
138     }
139
140     // Update reward variables for all pools. Be careful of gas spending!
141     function massUpdatePools() public {
142         uint256 length = poolInfo.length;
143         for (uint256 pid = 0; pid < length; ++pid) {
144             updatePool(pid);
145         }
146     }
147
148     // Update reward variables of the given pool to be up-to-date.
149     function updatePool(uint256 _pid) public {
150         PoolInfo storage pool = poolInfo[_pid];
151         if (block.number <= pool.lastRewardBlock) {
152             return;
153         }
154         uint256 lpSupply = pool.lpToken.balanceOf(address(this));
```

```
155         if (lpSupply == 0 || pool.allocPoint == 0) {
156             pool.lastRewardBlock = block.number;
157             return;
158         }
159         uint256 multiplier = getMultiplier(pool.lastRewardBlock, block.number);
160         uint256 fishReward = multiplier.mul(fishPerBlock).mul(pool.allocPoint).div(totalAl
161         fish.mint(devAddress, fishReward.div(10));
162         fish.mint(address(this), fishReward);
163         pool.accFishPerShare = pool.accFishPerShare.add(fishReward.mul(1e18).div(lpSupply)
164         pool.lastRewardBlock = block.number;
165     }
166 
167     // Deposit LP tokens to MasterChef for FISH allocation.
168     function deposit(uint256 _pid, uint256 _amount, address _referrer) public nonReentrant
169         PoolInfo storage pool = poolInfo[_pid];
170         UserInfo storage user = userInfo[_pid][msg.sender];
171         updatePool(_pid);
172         if (_amount > 0 && address(referral) != address(0) && _referrer != address(0) && _
173             referral.recordReferral(msg.sender, _referrer);
174         }
175         if (user.amount > 0) {
176             uint256 pending = user.amount.mul(pool.accFishPerShare).div(1e18).sub(user.rewa
177             if (pending > 0) {
178                 safeFishTransfer(msg.sender, pending);
179                 payReferralCommission(msg.sender, pending);
180             }
181         }
182         if (_amount > 0) {
183             pool.lpToken.safeTransferFrom(address(msg.sender), address(this), _amount);
184             if (pool.depositFeeBP > 0) {
185                 uint256 depositFee = _amount.mul(pool.depositFeeBP).div(10000);
186                 pool.lpToken.safeTransfer(feeAddress, depositFee.div(2));
187                 pool.lpToken.safeTransfer(vaultAddress, depositFee.div(2));
188                 user.amount = user.amount.add(_amount).sub(depositFee);
189             } else {
190                 user.amount = user.amount.add(_amount);
191             }
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complex
Solidity	43	6443	1150	663	4630	376

Comments to Code 663/4630 = 14%

Javascript Tests

Language	Files	Lines	Blanks	Comments	Code	Complex
JavaScript	0	0	0	0	0	0

Tests to Code 0/0 = 0%