# 0.7

## Snowball Finance Process Quality Review

Score: 38%

## Overview

This is a Snowball Finance Process Quality Review completed on July 29th 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

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:** Avalanche

**Guidance:**

Ethereum
Binance Smart Chain
Polygon
Avalanche

---

## 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://snowballs.gitbook.io/snowball-docs/resources/smart-contracts/snowball-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 over 10 transactions a day on contract *SwapFlashLoan.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/Snowball-Finance/protocol.

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 259 commits and 6 branches, this is a healthy 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

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

> ⚠ **Answer:** No

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://snowballs.gitbook.io/snowball-docs/.

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

> ⚠ **Answer:** N

There are no basic software functions documented in the Snowball Finance 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 documented in the Snowball Finance 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 13% 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%

There are no software functions documented in the Snowball Finance documentation.

**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 29% 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:** 0%

There is no evidence of code coverage in any of the Snowball Finance GitHub repositories.

**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/Snowball-Finance/protocol/tree/master/scripts.

**14) Report of the results (%)**

> ⚠ **Answer:** 0%

No test results report has been found in any of the Snowball Finance documentation.

**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%

No evidence of a Snowball Finance Formal Verification test has been found.

**16) Stress Testing environment (%)**

> ⚠ **Answer:** 0%

No evidence of Snowball Finance's test-net smart contract usage was found in their documentation.

---

## 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:** 20%

There has been no audit performed yet on Snowball Finance.

**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%

Snowball Finance's Bug Bounty Program is active and offers participants up to 100k in rewards for the most critical of bug 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:** 100%

Snowball Finance's access control docs can easily be found in the "Governance" section of their 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:** 60%

a) Contracts are clearly labelled as upgradeable through Snowball Finance's voting mechanism.

b) There are 5 MultiSig holders.

**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%

All of the Snowball Finance governance information is written in user-friendly language.

**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%

Snowball Finance has a *Pausable.sol* contract in their repository which essentially allows them to pause contracts whenever they want.

**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 | Snowball Finance | |
|---|---|---|---|
| **PQ Audit Scoring Matrix (v0.7)** | **Points** | **Answer** | **Points** |
| Total | **260** | | **98** |

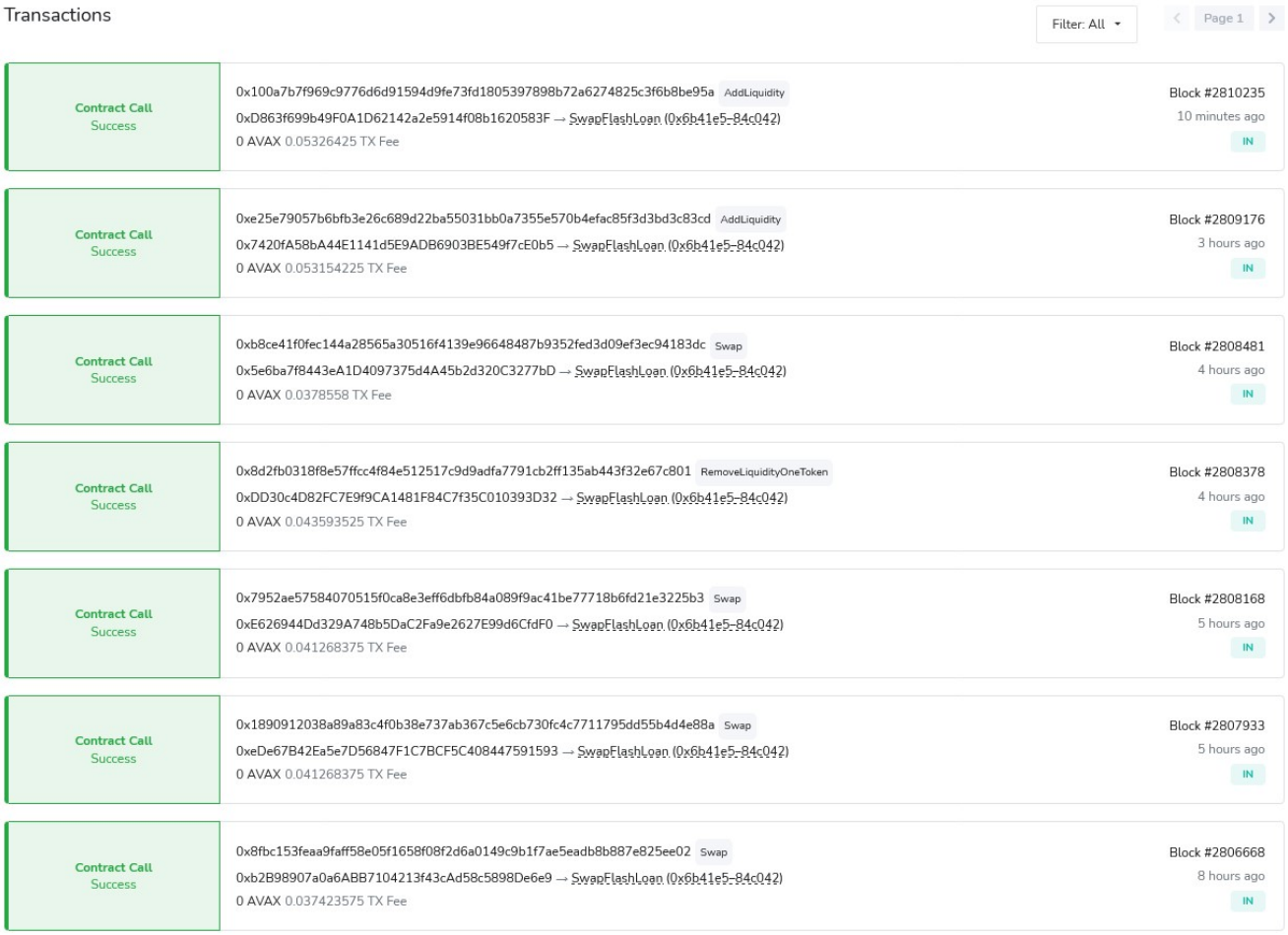
| **Code and Team** | | | **38%** |
|---|---|---|---|
| 1) Are the executing code addresses readily available? (%) | 20 | 100% | 20 |
| 2) Is the code actively being used? (%) | 5 | 100% | 5 |
| 3) Is there a public software repository? (Y/N) | 5 | Y | 5 |
| 4) Is there a development history visible? (%) | 5 | 100% | 5 |
| 5) Is the team public (not anonymous)? (Y/N) | 15 | 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 | 40% | 8 |
| 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 | Y | 5 |
| 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 | 20% | 14 |
| 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 controls | 5 | 100% | 5 |
| 20) Is the information clear and complete | 10 | 60% | 6 |
| 21) Is the information in non-technical terms | 10 | 90% | 9 |
| 22) Is there Pause Control documentation including records of tests | 10 | 40% | 4 |
| | | | |
| **Section Scoring** | | | |
| Code and Team | 50 | 70% | |
| Documentation | 45 | 11% | |
| Testing | 50 | 26% | |
| Security | 80 | 26% | |
| Access Controls | 35 | 69% | |

### Executing Code Appendix

| Role | Address |
|---|---|
| `Snowball Token (SNOB)` | 0xc38f41a296a4493ff429f1238e030924a1542e50 |
| `Governance` | 0xFdd994AD468cd39a4a3a3C3A0c460BB2213159B6 |
| `Treasury` | 0x294aB3200ef36200db84C4128b7f1b4eec71E38a |
| `Proposal 3 Funds` | 0x5df42ace37bA4AceB1f3465Aad9bbAcaA238D652 |
| `Controller` | 0xf7B8D9f8a82a7a6dd448398aFC5c77744Bd6cb85 |
| `Staked SNOB (xSNOB)` | 0x83952E7ab4aca74ca96217D6F8f7591BEaD6D64E |
| `Gauge Proxy` | 0xFc371bA1E7874Ad893408D7B581F3c8471F03D2C |

| | |
|---|---|
| IceQueen (Retired) | 0xB12531a2d758c7a8BF09f44FC88E646E1BF9D375 |

## Code Used Appendix




## Example Code Appendix

```
1
2 contract ControllerV4 {
3     using SafeERC20 for IERC20;
4     using Address for address;
5     using SafeMath for uint256;
6
7     address public constant burn = 0x000000000000000000000000000000000000dEaD;
8     address public onesplit = 0xC586BeF4a0992C495Cf22e1aeEE4E446CECDee0E;
9
10    address public governance;
11    address public strategist;
12    address public devfund;
13    address public treasury;
14    address public timelock;
15
16
```

```
    // Convenience fee 0.1%
    uint256 public convenienceFee = 100;
    uint256 public constant convenienceFeeMax = 100000;

    mapping(address => address) public globes; // takes lp address and returns associated g
    mapping(address => address) public strategies; // takes lp and returns associated strat
    mapping(address => mapping(address => address)) public converters;
    mapping(address => mapping(address => bool)) public approvedStrategies;
    mapping(address => bool) public approvedGlobeConverters;

    uint256 public split = 500;
    uint256 public constant max = 10000;

    constructor(
        address _governance,
        address _strategist,
        address _timelock,
        address _devfund,
        address _treasury
    ) public {
        governance = _governance;
        strategist = _strategist;
        timelock = _timelock;
        devfund = _devfund;
        treasury = _treasury;
    }

    function setDevFund(address _devfund) public {
        require(msg.sender == governance, "!governance");
        devfund = _devfund;
    }

    function setTreasury(address _treasury) public {
        require(msg.sender == governance, "!governance");
        treasury = _treasury;
    }

    function setStrategist(address _strategist) public {
        require(msg.sender == governance, "!governance");
        strategist = _strategist;
    }

    function setSplit(uint256 _split) public {
        require(msg.sender == governance, "!governance");
        require(_split <= max, "numerator cannot be greater than denominator");
        split = _split;
    }

    function setOneSplit(address _onesplit) public {
        require(msg.sender == governance, "!governance");
        onesplit = _onesplit;
    }

```

```
69
    function setGovernance(address _governance) public {
70      require(msg.sender == governance, "!governance");
71      governance = _governance;
72  }
73
74  function setTimelock(address _timelock) public {
75      require(msg.sender == timelock, "!timelock");
76      timelock = _timelock;
77  }
78
79  function setGlobe(address _token, address _globe) public {
80      require(
81          msg.sender == strategist || msg.sender == governance,
82          "!strategist"
83      );
84      require(globes[_token] == address(0), "globe");
85      globes[_token] = _globe;
86  }
87
88  function approveGlobeConverter(address _converter) public {
89      require(msg.sender == governance, "!governance");
90      approvedGlobeConverters[_converter] = true;
91  }
92
93  function revokeGlobeConverter(address _converter) public {
94      require(msg.sender == governance, "!governance");
95      approvedGlobeConverters[_converter] = false;
96  }
97
98  function approveStrategy(address _token, address _strategy) public {
99      require(msg.sender == timelock, "!timelock");
100     approvedStrategies[_token][_strategy] = true;
101 }
102
103 function revokeStrategy(address _token, address _strategy) public {
104     require(msg.sender == governance, "!governance");
105     require(strategies[_token] != _strategy, "cannot revoke active strategy");
106     approvedStrategies[_token][_strategy] = false;
107 }
108
109 function setConvenienceFee(uint256 _convenienceFee) external {
110     require(msg.sender == timelock, "!timelock");
111     convenienceFee = _convenienceFee;
112 }
113
114 function setStrategy(address _token, address _strategy) public {
115     require(
116         msg.sender == strategist || msg.sender == governance,
117         "!strategist"
118     );
119     require(approvedStrategies[_token][_strategy] == true, "!approved");
120
```

```
        address _current = strategies[_token];
        if (_current != address(0)) {
            IStrategy(_current).withdrawAll();
        }
        strategies[_token] = _strategy;
    }

    function earn(address _token, uint256 _amount) public {
        address _strategy = strategies[_token];
        address _want = IStrategy(_strategy).want();
        if (_want != _token) {
            address converter = converters[_token][_want];
            IERC20(_token).safeTransfer(converter, _amount);
            _amount = Converter(converter).convert(_strategy);
            IERC20(_want).safeTransfer(_strategy, _amount);
        } else {
            IERC20(_token).safeTransfer(_strategy, _amount);
        }
        IStrategy(_strategy).deposit();
    }

    function balanceOf(address _token) external view returns (uint256) {
        return IStrategy(strategies[_token]).balanceOf();
    }

    function withdrawAll(address _token) public {
        require(
            msg.sender == strategist || msg.sender == governance,
            "!strategist"
        );
        IStrategy(strategies[_token]).withdrawAll();
    }

    function inCaseTokensGetStuck(address _token, uint256 _amount) public {
        require(
            msg.sender == strategist || msg.sender == governance,
            "!governance"
        );
        IERC20(_token).safeTransfer(msg.sender, _amount);
    }

    function inCaseStrategyTokenGetStuck(address _strategy, address _token)
        public
    {
        require(
            msg.sender == strategist || msg.sender == governance,
            "!governance"
        );
        IStrategy(_strategy).withdraw(_token);
    }

    function getExpectedReturn(
        address _strategy,
        address _token
```

```
        address _token,
        uint256 parts
    ) public view returns (uint256 expected) {
        uint256 _balance = IERC20(_token).balanceOf(_strategy);
        address _want = IStrategy(_strategy).want();
        (expected, ) = OneSplitAudit(onesplit).getExpectedReturn(
            _token,
            _want,
            _balance,
            parts,
            0
        );
    }

    // Only allows to withdraw non-core strategy tokens ~ this is over and above normal yi
    function yearn(
        address _strategy,
        address _token,
        uint256 parts
    ) public {
        require(
            msg.sender == strategist || msg.sender == governance,
            "!governance"
        );
```

**SLOC Appendix**

Solidity Contracts

| Language | Files | Lines | Blanks | Comments | Code | Complex |
|---|---|---|---|---|---|---|
| Solidity | 10 | 1731 | 264 | 171 | 1296 | 141 |

Comments to Code 171/1296 = 13%

Javascript Tests

| Language | Files | Lines | Blanks | Comments | Code | Complex |
|---|---|---|---|---|---|---|
| JavaScript | 5 | 560 | 127 | 64 | 369 | 0 |

Tests to Code 369/1296 = 29%