0.7

HoneyFarm 2.0 Process Quality Review

Score: 42%

Overview

This is a HoneyFarm 2.0 Process Quality Review completed on October 27th 2021. It was performed using the Process Review process (version 0.7.3) and is documented here. The review was performed by David of DeFiSafety. Check out our Telegram.

The final score of the review is **42%**, 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 blockchains used by this protocol. This report covers all of the blockchains upon which the protocol is deployed.

> **Chain:** Binance Smart Chain

Guidance:

Ethereum
Binance Smart Chain
Polygon
Avalanche
Terra
Celo
Arbitrum
Solana

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 on the website https://bee.honeyfarm.finance, as indicated in the Appendix. Documentation is clearly labelled and found in the "Docs" element in their navigation bar. This will redirect you to their GitBook page.

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 2936 transactions a day on contract 0x88E21dedEf04cf24AFe1847B0F6927a719AA8F35, as indicated in the Appendix.

Note: Transactions are not well labelled but can be found by clicking on $BEE dollar amount in the bottom left corner of the navigation bar. This will redirect you to their BscScan page in a new tab.

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/HoneyFarmFi/HoneyFarmContracts

The HoneyFarm repository is hidden under the Social Networks element of the GitBook page, yet is well labelled once found.

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%

HoneyFarm contracts has one branch with six commits.

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

Location: Not made available despite having a team section.

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.honeyfarm.finance/

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

> ⚠️ **Answer:** No

No software function documentation available.

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%

No software function documentation available.

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:** 100%

Code examples are in the Appendix. As per the SLOC, there is 109% 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

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

> ⚠ **Answer:** 0%

No evidence of function documentation to be found.

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%

Code examples are in the Appendix. As per the SLOC, there is 0% testing to code (TtC). This figure is due entirely to the lack of availability or evidence that any testing was conducted.

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%

No evidence of testing to be found

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:** 0%

No evidence of testing to be found.

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%

No evidence of testing to be found.

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 testing to be found.

16) Stress Testing environment (%)

> ⚠ **Answer:** 0%

No evidence of testing to be found.

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%

One audit was performed by Peckshield on the 16th of October 2021. The audit reported 4 issues; 2 low severity, and 2 medium severity. 1 medium severity and 1 low severity issues were fixed, and the remaining were confirmed.

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:** 50%

HoneyFarm has an active bug bounty program with Immunefi. They offer a maximum payout of $50 000. This represents 0.2% of HoneyFarm's TVL ($20,883,070).

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%

No information available on access controls.

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%

No information on governance available.

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%

Information not available.

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%

Information not available.

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	HoneyFarm 2.0	
PQ Audit Scoring Matrix (v0.7)	**Points**	**Answer**	**Points**
Total	**260**		102
Code and Team			**39%**
1) Are the executing code addresses readily available? (%)	**20**	70%	14
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**	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**	100%	5
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**	n	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**	50%	5
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

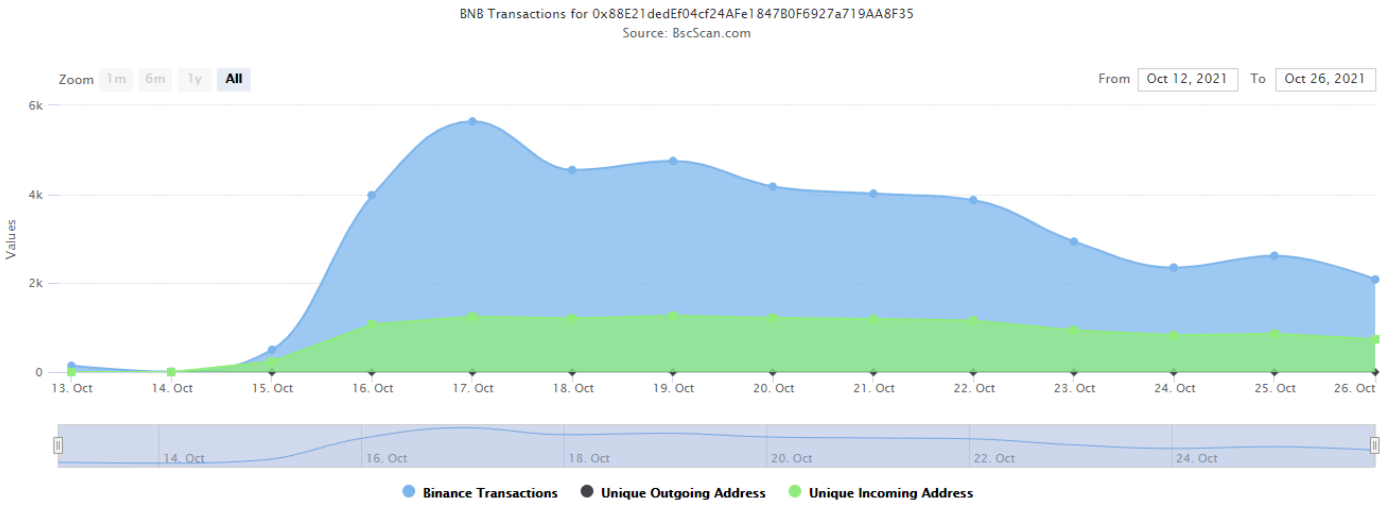
Executing Code Appendix

MasterChef: 0x88E21dedEf04cf24AFe1847B0F6927a719AA8F35

BEE Token: 0x1A8d7AC01d21991BF5249A3657C97b2B6d919222

Timelock:

Code Used Appendix



Example Code Appendix

```
// SPDX-License-Identifier: MIT


// file "./IBEP20.sol";

pragma solidity >=0.4.0;

interface IBEP20 {
    /**
     * @dev Returns the amount of tokens in existence.
     */
    function totalSupply() external view returns (uint256);

    /**
     * @dev Returns the token decimals.
     */
    function decimals() external view returns (uint8);

    /**
     * @dev Returns the token symbol.
     */
    function symbol() external view returns (string memory);

    /**
     * @dev Returns the token name.
     */
    function name() external view returns (string memory);
```

```
28
29      /**
30       * @dev Returns the bep token owner.
31       */
32      function getOwner() external view returns (address);
33
34      /**
35       * @dev Returns the amount of tokens owned by `account`.
36       */
37      function balanceOf(address account) external view returns (uint256);
38
39      /**
40       * @dev Moves `amount` tokens from the caller's account to `recipient`.
41       *
42       * Returns a boolean value indicating whether the operation succeeded.
43       *
44       * Emits a {Transfer} event.
45       */
46      function transfer(address recipient, uint256 amount) external returns (bool);
47
48      /**
49       * @dev Returns the remaining number of tokens that `spender` will be
50       * allowed to spend on behalf of `owner` through {transferFrom}. This is
51       * zero by default.
52       *
53       * This value changes when {approve} or {transferFrom} are called.
54       */
55      function allowance(address _owner, address spender) external view returns (uint256);
56
57      /**
58       * @dev Sets `amount` as the allowance of `spender` over the caller's tokens.
59       *
60       * Returns a boolean value indicating whether the operation succeeded.
61       *
62       * IMPORTANT: Beware that changing an allowance with this method brings the risk
63       * that someone may use both the old and the new allowance by unfortunate
64       * transaction ordering. One possible solution to mitigate this race
65       * condition is to first reduce the spender's allowance to 0 and set the
66       * desired value afterwards:
67       * https://github.com/ethereum/EIPs/issues/20#issuecomment-263524729
68       *
69       * Emits an {Approval} event.
70       */
71      function approve(address spender, uint256 amount) external returns (bool);
72
73      /**
74       * @dev Moves `amount` tokens from `sender` to `recipient` using the
75       * allowance mechanism. `amount` is then deducted from the caller's
76       * allowance.
77       *
78       * Returns a boolean value indicating whether the operation succeeded.
79       *
80       * Emits a {Transfer} event.
```

```
     */
    function transferFrom(
        address sender,
        address recipient,
        uint256 amount
    ) external returns (bool);

    /**
     * @dev Emitted when `value` tokens are moved from one account (`from`) to
     * another (`to`).
     *
     * Note that `value` may be zero.
     */
    event Transfer(address indexed from, address indexed to, uint256 value);

    /**
     * @dev Emitted when the allowance of a `spender` for an `owner` is set by
     * a call to {approve}. `value` is the new allowance.
     */
    event Approval(address indexed owner, address indexed spender, uint256 value);
}


pragma solidity >=0.6.2 <0.8.0;

/**
 * @dev Collection of functions related to the address type
 */
library Address {
    /**
     * @dev Returns true if `account` is a contract.
     *
     * [IMPORTANT]
     * ====
     * It is unsafe to assume that an address for which this function returns
     * false is an externally-owned account (EOA) and not a contract.
     *
     * Among others, `isContract` will return false for the following
     * types of addresses:
     *
     *  - an externally-owned account
     *  - a contract in construction
     *  - an address where a contract will be created
     *  - an address where a contract lived, but was destroyed
     * ====
     */
    function isContract(address account) internal view returns (bool) {
        // This method relies on extcodesize, which returns 0 for contracts in
        // construction, since the code is only stored at the end of the
        // constructor execution.

        uint256 size;
        // solhint-disable-next-line no-inline-assembly
```

```
134
        assembly { size := extcodesize(account) }
135         return size > 0;
136     }
137
138     /**
139      * @dev Replacement for Solidity's `transfer`: sends `amount` wei to
140      * `recipient`, forwarding all available gas and reverting on errors.
141      *
142      * https://eips.ethereum.org/EIPS/eip-1884[EIP1884] increases the gas cost
143      * of certain opcodes, possibly making contracts go over the 2300 gas limit
144      * imposed by `transfer`, making them unable to receive funds via
145      * `transfer`. {sendValue} removes this limitation.
146      *
147      * https://diligence.consensys.net/posts/2019/09/stop-using-soliditys-transfer-now/[Lea
148      *
149      * IMPORTANT: because control is transferred to `recipient`, care must be
150      * taken to not create reentrancy vulnerabilities. Consider using
151      * {ReentrancyGuard} or the
152      * https://solidity.readthedocs.io/en/v0.5.11/security-considerations.html#use-the-chec
153      */
154     function sendValue(address payable recipient, uint256 amount) internal {
155         require(address(this).balance >= amount, "Address: insufficient balance");
156
157         // solhint-disable-next-line avoid-low-level-calls, avoid-call-value
158         (bool success, ) = recipient.call{ value: amount }("");
159         require(success, "Address: unable to send value, recipient may have reverted");
160     }
161
162     /**
163      * @dev Performs a Solidity function call using a low level `call`. A
164      * plain`call` is an unsafe replacement for a function call: use this
165      * function instead.
166      *
167      * If `target` reverts with a revert reason, it is bubbled up by this
168      * function (like regular Solidity function calls).
169      *
170      * Returns the raw returned data. To convert to the expected return value,
171      * use https://solidity.readthedocs.io/en/latest/units-and-global-variables.html?highl
172      *
173      * Requirements:
174      *
175      * - `target` must be a contract.
176      * - calling `target` with `data` must not revert.
177      *
178      * _Available since v3.1._
179      */
180     function functionCall(address target, bytes memory data) internal returns (bytes memory
181       return functionCall(target, data, "Address: low-level call failed");
182     }
183
184     /**
185      * @dev Same as {xref-Address-functionCall-address-bytes-}[`functionCall`], but with
```

```
186      *  errorMessage  as a fallback revert reason when  target  reverts.
187      *
188      * _Available since v3.1._
189      */
190     function functionCall(address target, bytes memory data, string memory errorMessage) i
191         return functionCallWithValue(target, data, 0, errorMessage);
192     }
193
194     /**
195      * @dev Same as {xref-Address-functionCall-address-bytes-}[`functionCall`],
196      * but also transferring `value` wei to `target`.
197      *
198      * Requirements:
199      *
200      * - the calling contract must have an ETH balance of at least `value`.
201      * - the called Solidity function must be `payable`.
202      *7
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complex
Solidity	4	5077	649	2306	2122	222

Comments to Code 2306/2122 = 109%

Javascript Tests

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

Tests to Code n/a