0.7

Tranchess Process Quality Review

Score: 66%

Overview

This is a Tranchess Process Quality Review completed on September 7th 2021. It was performed using the Process Review process (version 0.7.3) and is documented here. The review was performed by Mawuli of DeFiSafety. Check out our Telegram.

The final score of the review is **66%**, a **Close 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:** 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:** 100%

They are available at website https://docs.tranchess.com/faq/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 *TransparentUpradblepProxy.sol* (swap contract), 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/tranchess/contract-core.

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 364 commits and 4 branches, this is a robust 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://docs.tranchess.com/whitepaper.

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

> ✓ **Answer:** Y

There are software functions documented in the Tranchess documentation at https://github.com/tranchess/contract-core/blob/main/spec

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:** 100%

The software spec at https://github.com/tranchess/contract-core/blob/main/spec/README_EXCHANGE.md is complete, readable and detailed.

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:** 24%

Code examples are in the Appendix. As per the SLOC, there is 24% 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:** 40%

There are no software functions documented in the Tranchess documentation. Therefore, there is no traceability as to its implementation in their 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:** 100%

Code examples are in the Appendix. As per the SLOC, there is 310% 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

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

Answer: 50%

There is no evidence of code coverage in any of the Tranchess documentation, but there is a complete test suite.

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/tranchess/contract-core#readme.

14) Report of the results (%)

Answer: 0%

There is no evidence of a test result report in any of the Tranchess 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%

There is no evidence of a Formal Verification tests in an of the Tranchess documentation.

16) Stress Testing environment (%)

⚠ **Answer:** 0%

There is no evidence of any Tranchess testnet smart contract usage.

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%

Certik published a Tranchess audit report on June 2nd 2021, which is before their mainnet launch in July 2021.

PeckShield published a Tranchess audit on June 28th 2021, which is before their mainnet launch in July 2021.

Note: Fix recommendations were implemented by the Tranchess 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, (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%

Tranchess' Bug Bounty program rewards participating users with up to 50K 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:** 0%

There are currently no access controls documented in the Tranchess 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 currently no access controls documented in the Tranchess 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 are currently no access controls documented in the Tranchess 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 a Pause Control or a similar function in the Tranchess documentation.

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	Tranchess	
PQ Audit Scoring Matrix (v0.7)	Points	Answer	Points

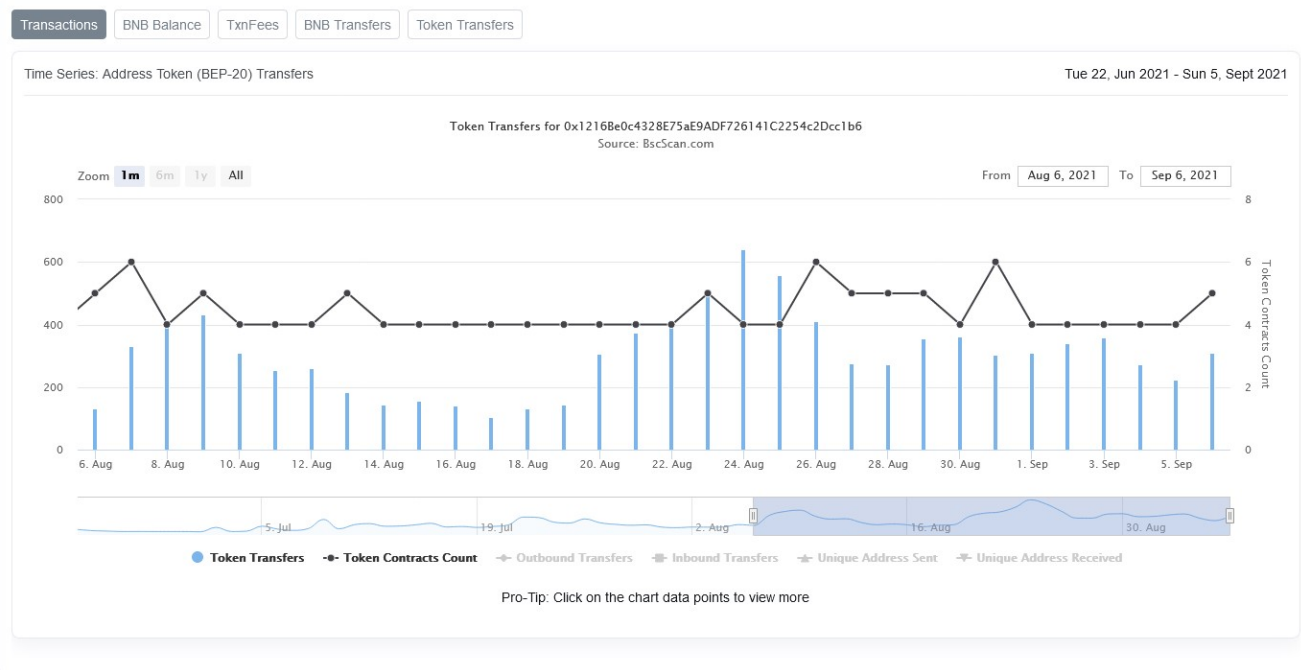
Total	260		172.7
Code and Team			**66%**
1) Are the executing code addresses readily available? (%)	20	100%	20
2) Is the code actively being used? (%)	5	100%	5
3) Is there a public software repository? (Y/N)	5	Y	5
4) Is there a development history visible? (%)	5	100%	5
5) Is the team public (not anonymous)? (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	Y	10
8) Does the software function documentation fully (100%) cov	15	100%	15
9) Are there sufficiently detailed comments for all functions w	5	24%	1.2
10) Is it possible to trace from software documentation to the	10	40%	4
Testing			
11) Full test suite (Covers all the deployed code) (%)	20	100%	20
12) Code coverage (Covers all the deployed lines of code, or ex	5	50%	2.5
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	0%	0
16) Stress Testing environment (%)	5	0%	0
Security			
17) Did 3rd Party audits take place? (%)	70	100%	70
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	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	10	0%	0
Section Scoring			
Code and Team	50	70%	
Documentation	45	78%	
Testing	50	55%	
Security	80	94%	
Access Controls	35	0%	

Executing Code Appendix

Name	Address
TwapOracle	0xD924955C92a67d32e0C10e686a6003B22eCFba28
AprOracle	0x8424D933FbB73665E5a8880de63C7B1366a56EeD
Chess	0x20de22029ab63cf9A7Cf5fEB2b737Ca1eE4c82A6
ChessSchedule	0xf071dE0E7A6FfCeee252Df25678c725f04A03b80
TimelockController	0x4BB3AeB5Ba75bC6A44177907B54911b19d1cF8f7
ProxyAdmin	0x88C8890505384F4Eb3A281274b1DEdFFf8448147

Fund	0xd6B3B86209eBb3C608f3F42Bf52818169944E402
Token QUEEN	0x15D0318Fddf785aC0d3ba690C0033b3bedF4c648
Token BISHOP	0x8cC456B384C8aD06BF430F4F130Aa63EF0dc6f85
Token ROOK	0x80da8Ca6c3DabD3a9f06Ca8eEed5d61687fab7ef
PrimaryMarket	0x19Ca3baAEAf37b857026dfEd3A0Ba63987A1008D
Swap	0x1216Be0c4328E75aE9ADF726141C2254c2Dcc1b6
VotingEscrow	0x95A2bBCD64E2859d40E2Ad1B5ba49Dc0e1Abc6C2
FeeDistributor	0x85ae5e9d510d8723438b0135CBf29d4F2E8BCda8
InterestRateBallot	0xeB76E34834fB0E2c31D92f0284466385bcE5c09A

Code Used Appendix



Example Code Appendix

```
contract TransparentUpgradeableProxy is UpgradeableProxy {
    /**
     * @dev Initializes an upgradeable proxy managed by `_admin`, backed by the implementa
     * optionally initialized with `_data` as explained in {UpgradeableProxy-constructor}.
     */
    constructor(address _logic, address admin_, bytes memory _data) public payable Upgrade
        assert(_ADMIN_SLOT == bytes32(uint256(keccak256("eip1967.proxy.admin")) - 1));

```

```
        _setAdmin(admin_);
    }

    /**
     * @dev Emitted when the admin account has changed.
     */
    event AdminChanged(address previousAdmin, address newAdmin);

    /**
     * @dev Storage slot with the admin of the contract.
     * This is the keccak-256 hash of "eip1967.proxy.admin" subtracted by 1, and is
     * validated in the constructor.
     */
    bytes32 private constant _ADMIN_SLOT = 0xb53127684a568b3173ae13b9f8a6016e243e63b6e8ee1

    /**
     * @dev Modifier used internally that will delegate the call to the implementation unl
     */
    modifier ifAdmin() {
        if (msg.sender == _admin()) {
            _;
        } else {
            _fallback();
        }
    }

    /**
     * @dev Returns the current admin.
     *
     * NOTE: Only the admin can call this function. See {ProxyAdmin-getProxyAdmin}.
     *
     * TIP: To get this value clients can read directly from the storage slot shown below
     * https://eth.wiki/json-rpc/API#eth_getstorageat[`eth_getStorageAt`] RPC call.
     * `0xb53127684a568b3173ae13b9f8a6016e243e63b6e8ee1178d6a717850b5d6103`
     */
    function admin() external ifAdmin returns (address admin_) {
        admin_ = _admin();
    }

    /**
     * @dev Returns the current implementation.
     *
     * NOTE: Only the admin can call this function. See {ProxyAdmin-getProxyImplementation}
     *
     * TIP: To get this value clients can read directly from the storage slot shown below
     * https://eth.wiki/json-rpc/API#eth_getstorageat[`eth_getStorageAt`] RPC call.
     * `0x360894a13ba1a3210667c828492db98dca3e2076cc3735a920a3ca505d382bbc`
     */
    function implementation() external ifAdmin returns (address implementation_) {
        implementation_ = _implementation();
    }

    /**
```

```
61
         * @dev Changes the admin of the proxy.
62       *
63       * Emits an {AdminChanged} event.
64       *
65       * NOTE: Only the admin can call this function. See {ProxyAdmin-changeProxyAdmin}.
66       */
67      function changeAdmin(address newAdmin) external ifAdmin {
68          require(newAdmin != address(0), "TransparentUpgradeableProxy: new admin is the zer
69          emit AdminChanged(_admin(), newAdmin);
70          _setAdmin(newAdmin);
71      }
72
73      /**
74       * @dev Upgrade the implementation of the proxy.
75       *
76       * NOTE: Only the admin can call this function. See {ProxyAdmin-upgrade}.
77       */
78      function upgradeTo(address newImplementation) external ifAdmin {
79          _upgradeTo(newImplementation);
80      }
81
82      /**
83       * @dev Upgrade the implementation of the proxy, and then call a function from the new
84       * by `data`, which should be an encoded function call. This is useful to initialize n
85       * proxied contract.
86       *
87       * NOTE: Only the admin can call this function. See {ProxyAdmin-upgradeAndCall}.
88       */
89      function upgradeToAndCall(address newImplementation, bytes calldata data) external pay
90          _upgradeTo(newImplementation);
91          // solhint-disable-next-line avoid-low-level-calls
92          (bool success,) = newImplementation.delegatecall(data);
93          require(success);
94      }
95
96      /**
97       * @dev Returns the current admin.
98       */
99      function _admin() internal view returns (address adm) {
100         bytes32 slot = _ADMIN_SLOT;
101         // solhint-disable-next-line no-inline-assembly
102         assembly {
103             adm := sload(slot)
104         }
105     }
106
107     /**
108      * @dev Stores a new address in the EIP1967 admin slot.
109      */
110     function _setAdmin(address newAdmin) private {
111         bytes32 slot = _ADMIN_SLOT;
112
```

```
113         // solhint-disable-next-line no-inline-assembly
114         assembly {
115             sstore(slot, newAdmin)
116         }
117     }
118
119     /**
120      * @dev Makes sure the admin cannot access the fallback function. See {Proxy-_beforeFa
121      */
122     function _beforeFallback() internal override virtual {
123         require(msg.sender != _admin(), "TransparentUpgradeableProxy: admin cannot fallbac
124         super._beforeFallback();
125     }
126 }
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complex
Solidity	18	3702	463	616	2623	366

Comments to Code 616/2623 = 24%

Javascript Tests

Language	Files	Lines	Blanks	Comments	Code	Complex
TypeScript	15	9436	1074	328	8034	58
Solidity	3	120	20	3	97	1

Tests to Code 8131/2623 = 310%