0.7

Jetfuel Finance Process Quality Review

Score: 38%

Overview

This is a Jetfuel Finance Process Quality Review completed on July 1st 2021. It was performed using the Process Review process (version 0.7.2) 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.

This completed report is copyright (c) DeFiSafety 2021. Permission is given to copy in whole, retaining this copyright label.

Chain

This section indicates the blockchain used by this protocol.

> ✓ **Chain: Binance Smart Chain**

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 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.jetfuel.finance/more-information/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 labelling not clear or easy to find
0% Executing addresses could not be found

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

> ✓ Answer: 100%

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

Percentage Score 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: No

Although the devs have a Public Software Repository, they do not actively develop on it. The devs develop on a Private Software Repository.

GitHub (defunct): https://github.com/jetfuelfinance

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

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

> ⚠ Answer: 0%

With a private repository, it is impossible to verify the development history of the protocol.

This 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

No Jetfuel Finance team or developer info was found.

For a yes in this question the real names of some team members must be public on the website or other documentation. If the team is anonymous and 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.jetfuel.finance/

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

> ⚠ Answer: No

There are no basic software functions documented in Jetfuel Finance's 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 Jetfuel Finance's 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 improve by adding content to the requirements 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: 90%

Code examples are in the Appendix. As per the SLOC, there is 92% 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 Jetfuel Finance's documentation, therefore it is not possible to trace them to their implementation in 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 requirements to code such that it is clear where each requirement is coded. 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%

With no public repository, it is impossible to verify if they have done any testing on the protocol.

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 improve 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 testing having been done in any of the JetFuel Finance audits or in their documentation.

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 improve by adding tests achieving 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

There is no information on how to run the tests.

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 a code coverage results report.

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 any Jetfuel Finance Formal Verification testing in any of their documentation or in web searches.

16) Stress Testing environment (%)

⚠ Answer: 0%

There is no evidence of any test-net smart contract usage in any of Jetfuel Finance's 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: 90%

Ether Authority published a Jetfuel Finance audit report on March 8th 2021.

VidarTheAuditor published a Jetfuel Finance audit report on January 12th 2021.

VidarTheAuditor published a Jetfuel Finance audit report on December 15th 2020.

Note 1: Jetfuel Finance launched on December 6th 2020.

Note 2: Most fix recommendations were implemented by the Jetfuel Finance 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

18) Is the bounty value acceptably high (%)

> ⚠ Answer: 0%

There is no evidence of a Jetfuel Finance Bug Bounty program 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

Active program means a third party actively driving hackers to the site. Inactive program would be static mention 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?
2`) 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 evidence of any admin access control information in any of Jetfuel Finance's 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 is no evidence of any access control information in any of Jetfuel Finance's 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 evidence of any access control information in any of Jetfuel Finance's 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 similar function listed in any of Jetfuel Finance's documentation and 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

PQ Audit Scoring Matrix (v0.7)	Total Points	Jetfuel Finance Answer	Jetfuel Finance Points
Total	260		97.5
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	N	0
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	90%	4.5
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	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	50%	
Documentation	45	21%	
Testing	50	0%	
Security	80	79%	
Access Controls	35	0%	

Executing Code Appendix

Contracts

MasterFUEL:
https://bscscan.com/address/0x86f4bC1EBf2C209D12d3587B7085aEA5707d4B56

FUEL token:
https://bscscan.com/address/0x2090c8295769791ab7A3CF1CC6e0AA19F35e441A

JETS token:

https://bscscan.com/address/0xf6488205957f0b4497053d6422F49e27944eE3Dd

GFC token:
https://bscscan.com/token/0x94BaBBE728D9411612Ee41b20241a6FA251b26Ce

---

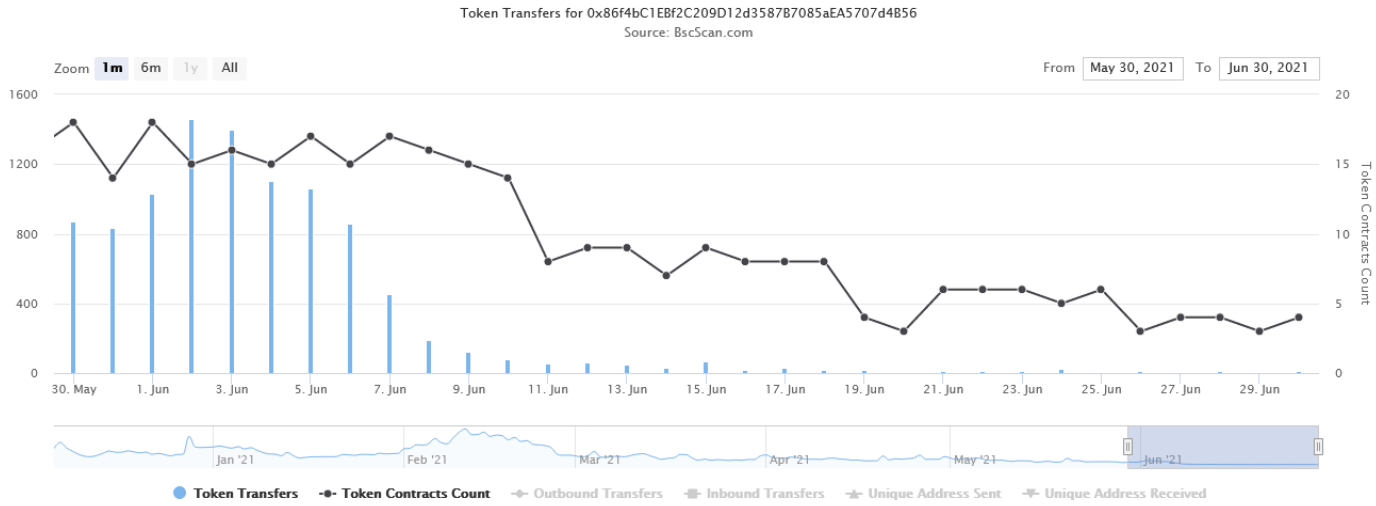
Token Addresses

FUEL: https://bscscan.com/address/0x2090c8295769791ab7A3CF1CC6e0AA19F35e441A
0x2090c8295769791ab7A3CF1CC6e0AA19F35e441A

JETS: https://bscscan.com/address/0xf6488205957f0b4497053d6422F49e27944eE3Dd
0xf6488205957f0b4497053d6422F49e27944eE3Dd

GFC: https://bscscan.com/token/0x94babbe728d9411612ee41b20241a6fa251b26ce
0x94babbe728d9411612ee41b20241a6fa251b26ce

Code Used Appendix



Example Code Appendix

```
    // View function to see pending FUEL on frontend.
    function pendingFuel(uint256 _pid, address _user) external view returns (uint256) {
        PoolInfo storage pool = poolInfo[_pid];
        UserInfo storage user = userInfo[_pid][_user];
        uint256 accFuelPerShare = pool.accFuelPerShare;
        uint256 lpSupply = pool.token.balanceOf(address(this));
        if (block.number > pool.lastRewardBlock && lpSupply != 0 && pool.lastRewardBlock <
            uint256 totalReward = getTotalRewardInfo(pool.lastRewardBlock, block.number);
            uint256 fuelReward = totalReward.mul(pool.allocPoint).div(totalAllocPoint);

```

```
            accFuelPerShare = accFuelPerShare.add(fuelReward.mul(accFuelPerShareMultiple).
        }
        return user.amount.mul(accFuelPerShare).div(accFuelPerShareMultiple).sub(user.rewa
    }

    // Update reward vairables for all pools. Be careful of gas spending!
    function massUpdatePools() public {
        uint256 length = poolInfo.length;
        for (uint256 pid = 0; pid < length; ++pid) {
            updatePool(pid);
        }
    }

    // Update reward variables of the given pool to be up-to-date.
    function updatePool(uint256 _pid) public {
        PoolInfo storage pool = poolInfo[_pid];
        if (block.number <= pool.lastRewardBlock) {
            return;
        }
        uint256 lpSupply = pool.token.balanceOf(address(this));
        if (lpSupply == 0) {
            pool.lastRewardBlock = block.number;
            return;
        }
        if (pool.lastRewardBlock >= maxRewardBlockNumber) {
            return;
        }
        uint256 totalReward = getTotalRewardInfo(pool.lastRewardBlock, block.number);
        uint256 fuelReward = totalReward.mul(pool.allocPoint).div(totalAllocPoint);
        fuel.mintTo(devAddr, fuelReward.div(40)); // 2.5% fuel devs fund
        fuel.mintTo(address(this), fuelReward);
        pool.accFuelPerShare = pool.accFuelPerShare.add(fuelReward.mul(accFuelPerShareMult
        pool.lastRewardBlock = block.number;
    }

    // Deposit LP tokens for FUEL allocation.
    function deposit(uint256 _pid, uint256 _amount) public {
        PoolInfo storage pool = poolInfo[_pid];
        UserInfo storage user = userInfo[_pid][msg.sender];
        updatePool(_pid);
        if (user.amount > 0) {
            uint256 pending = user.amount.mul(pool.accFuelPerShare).div(accFuelPerShareMul
                user.rewardDebt
            );
            if (pending > 0) {
                safeFuelTransfer(msg.sender, pending);
            }
        }
        pool.token.safeTransferFrom(address(msg.sender), address(this), _amount);
        user.amount = user.amount.add(_amount);
        user.rewardDebt = user.amount.mul(pool.accFuelPerShare).div(accFuelPerShareMultipl
        emit Deposit(msg.sender, _pid, _amount);
    }
```

```
63
64      // Withdraw LP tokens
65      function withdraw(uint256 _pid, uint256 _amount) public {
66          PoolInfo storage pool = poolInfo[_pid];
67          UserInfo storage user = userInfo[_pid][msg.sender];
68          require(user.amount >= _amount, 'withdraw: not good');
69          updatePool(_pid);
70          uint256 pending = user.amount.mul(pool.accFuelPerShare).div(accFuelPerShareMultiple
71              user.rewardDebt
72          );
73          if (pending > 0) {
74              safeFuelTransfer(msg.sender, pending);
75          }
76          if (_amount > 0) {
77              user.amount = user.amount.sub(_amount);
78              pool.token.safeTransfer(address(msg.sender), _amount);
79          }
80          user.rewardDebt = user.amount.mul(pool.accFuelPerShare).div(accFuelPerShareMultiple
81          emit Withdraw(msg.sender, _pid, _amount);
82      }
83
84      // Withdraw without caring about rewards. EMERGENCY ONLY.
85      function emergencyWithdraw(uint256 _pid) public {
86          PoolInfo storage pool = poolInfo[_pid];
87          UserInfo storage user = userInfo[_pid][msg.sender];
88          pool.token.safeTransfer(address(msg.sender), user.amount);
89          emit EmergencyWithdraw(msg.sender, _pid, user.amount);
90          user.amount = 0;
91          user.rewardDebt = 0;
92      }
93
94      // Safe Fuel transfer function, just in case if rounding error causes pool to not have
95      function safeFuelTransfer(address _to, uint256 _amount) internal {
96          uint256 fuelBal = fuel.balanceOf(address(this));
97          if (_amount > fuelBal) {
98              fuel.transfer(_to, fuelBal);
99          } else {
100             fuel.transfer(_to, _amount);
101         }
102     }
103
104     // Update dev address by the previous dev.
105     function dev(address _devAddr) public {
106         require(msg.sender == devAddr, 'you no fuel dev: wut?');
107         devAddr = _devAddr;
108     }
109 }
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complex
Solidity	4	4729	509	2026	2194	250

Comments to Code 2026/2194 = 92%

Javascript Tests

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

Tests to Code 0/0 = 0%