0.7

Value DeFi PQ Review (0.5)

Score : 8%

This is a YFV Finance now renamed Value Defi Process Quality Review completed on 28 September 2020. It was performed using the Process Review process (version 0.5) and is documented here. The review was performed by ShinkaRex of Caliburn Consulting. Check out our Telegram.

The final score of the review is 8%, a clear fail. The breakdown of the scoring is in Scoring Appendix. If all the contract addresses were public, the score would be 35%, taking into account the audits.

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.

1. **Here is my smart contract on the blockchain**
2. **You can see it matches a software repository used to develop the code**
3. **Here is the documentation that explains what my smart contract does**
4. **Here are the tests I ran to verify my smart contract**
5. **Here are the audit(s) performed to review my code by third party experts**

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, 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. 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.

Executing Code Verification

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. Is the executing code address(s) readily available? (Y/N)
2. Is the code actively being used? (%)
3. Are the Contract(s) Verified/Verifiable? (Y/N)
4. Does the code match a tagged version in the code hosting platform? (%)
5. Is the software repository healthy? (%)

Is the executing code address(s) readily available? (Y/N)

⚠ Answer: No

Actually, some contracts are available while many others are not. Because a considerable number of important contracts do not appear to have addresses we have put the answer at NO. They're addresses for the four tokens and for the pools. The list of contracts in the GitHub and obviously required given the functionality (vaults etc.) indicates these contracts must be on the main somewhere. Therefore, their addresses are not public.

The mainnet addresses of the following contracts were not publicly available;

- YFV_Vote
- YFV_DevRewards
- NoMintRewardPool
- YFVController
- YFVStrategy
- ValueBank
- ValueVault
- YValueVaultMaster

This review will look at Address 0xC2D55CE14a8e04AEF9B6bCfD105079b63C6a0AC8 as indicated in the Appendix. This review only covers the contract YFVRewards.

How to improve this score

Make the ethereum addresses of the smart contract utilized by your application available on either your website or your github (in the README for instance). Ensure the address is up to date. This is a very important question wrt to the final score.

Is the code actively being used? (%)

> ✓ Answer: 100%

Activity is well in excess of 100 transactions a day, 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

Are the Contract(s) Verified/Verifiable? (Y/N)

> ✓ Answer: Yes

0xC2D55CE14a8e04AEF9B6bCfD105079b63C6a0AC8 is the Etherscan verified contract address.

How to improve this score

Ensure that the deployed code is verified as described in this article for Etherscan or ETHPM. Improving this score may require redeployment.

Does the code match a tagged version on a code hosting platform? (%)

> ⚠ Answer: 10%

Of the contracts I verified, the following matched with the GitHub; YFVStakeV2. YFN, vETH while ValueMasterPool, YFVRewards, YFV_Governance, YFVGovernanceVault did not match. Where they did not match, many code changes were evident.

Guidance:

100%	All code matches and Repository was clearly labelled
60 %	All code matches but no labelled repository. Repository was found manually
30%	Almost all code does match perfectly and repository was found manually
0%	Most matching Code could not be found

GitHub address : https://github.com/yfv-finance/audit and https://github.com/yfv-finance/vaults

Deployed contracts in the following file;

YFV_Deployed.rar 75KB
Binary

Matching Repository: there is only 1 branch

How to improve this score

Ensure there is a clearly labelled repository holding all the contracts, documentation and tests for the deployed code. Ensure an appropriately labeled tag exists corresponding to deployment dates. Release tags are clearly communicated.

Is development software repository healthy? (%)

⚠ Answer: 0%

The Vaults GitHub has 3 commits, 1 branch and no releases. The audits repo has even less. Neither seem to be used for development or even deployment.

How to improve this score

Ensure there is a clearly labelled repository holding all the contracts, documentation and tests for the deployed code. Continue to test and perform other verification activities after deployment, including routine maintenance updating to new releases of testing and deployment tools.

Documentation

This section looks at the software documentation. The document explaining these questions is here.

Required questions are;

1. Is there a whitepaper? (Y/N)
2. Are the basic application requirements documented? (Y/N)
3. Do the requirements fully (100%) cover the deployed contracts? (%)
4. Are there sufficiently detailed comments for all functions within the deployed contract code (%)
5. Is it possible to trace software requirements to the implementation in code (%)

Is there a whitepaper? (Y/N)

✓ Answer: Yes

While not really a whitepaper, there is at least an introductory Medium article.

Location: https://medium.com/@yfv.finance/yfv-bringing-true-value-to-yield-farming-bddc4edf889a

Are the basic application requirements documented? (Y/N)

> ⚠ Answer: No

There is no software documentation evident.

How to improve this score

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

Do the requirements fully (100%) cover the deployed contracts? (%)

> ⚠ Answer: 0%

This program appears to be a fork of Yearn Finance. In the audits repo readme there are differences files for the token, pool contract and rewards contract between Yearn and YFV. However the status of the other contracts is unknown.

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.

Are there sufficiently detailed comments for all functions within the deployed contract code (%)

> ! Answer: 30%

When we looked at the last contract of the files that were in the Deployed file enclosed above we found a rather low commenting ratio which drives to to a 30% score

Code examples are in the Appendix. As per the SLOC, there is 17% commenting to code.

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.

Is it possible to trace requirements to the implementation in code (%)

> ⚠ Answer: 0%

As there is no software documentation, tracing to the code is impossible.

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;

1. Full test suite (Covers all the deployed code) (%)
2. Code coverage (Covers all the deployed lines of code, or explains misses) (%)
3. Scripts and instructions to run the tests (Y/N)
4. Packaged with the deployed code (Y/N)
5. Report of the results (%)
6. Formal Verification test done (%)
7. Stress Testing environment (%)

Is there a Full test suite? (%)

> ⚠ Answer: 0%

There is no test directory or other traces of any tests.

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.

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

> ⚠ Answer: 0%

With no tests evident, there can be no coverage.

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.

Scripts and instructions to run the tests (Y/N)

> ⚠ Answer: N

No instructions visible.

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.

Packaged with the deployed code (Y/N)

> ⚠ Answer: No

No tests, so they cannot be packaged with the code.

How to improve this score

Improving this score requires redeployment of the code, with the tests. This score gives credit to those who test their code before deployment and release them together. If a developer adds tests after deployment they can gain full points for all test elements except this one.

Report of the results (%)

⚠ Answer: 0%

No report was visible. No surprise, as there were no tests.

How to improve this score

Add a report with the results. The test scripts should generate the report or elements of it.

Formal Verification test done (%)

⚠ Answer: 0%

No indication of formal verification tests.

Stress Testing environment (%)

⚠ Answer: 0%

No Evidence of a stress test environment being used.

Audits

ⓘ Answer: 50%

There are three audits for YFV. All appear to have taken place after deployment. There is no indication of implementing recommendations and one audit had an item that should have been addressed. For this reason a score between 70% and 20%, in this case 50%.

However many of the smart contract addresses are unknown, so we don't know what code is executing with respect to the audits. Therefor the score is 0.

Guidance:

1. Multiple Audits performed before deployment and results public and implemented or not required (100%)
2. Single audit performed before deployment and results public and implemented or not required (90%)
3. Audit(s) performed after deployment and no changes required. Audit report is public. (70%)
4. No audit performed (20%)

5. Audit Performed after deployment, existence is public, report is not public and no improvements deployed OR smart contract address' not found, question 1 (0%)

Appendices

Author Details

The author of this review is Rex of Caliburn Consulting.

Email : rex@defisafety.com Twitter : @defisafety

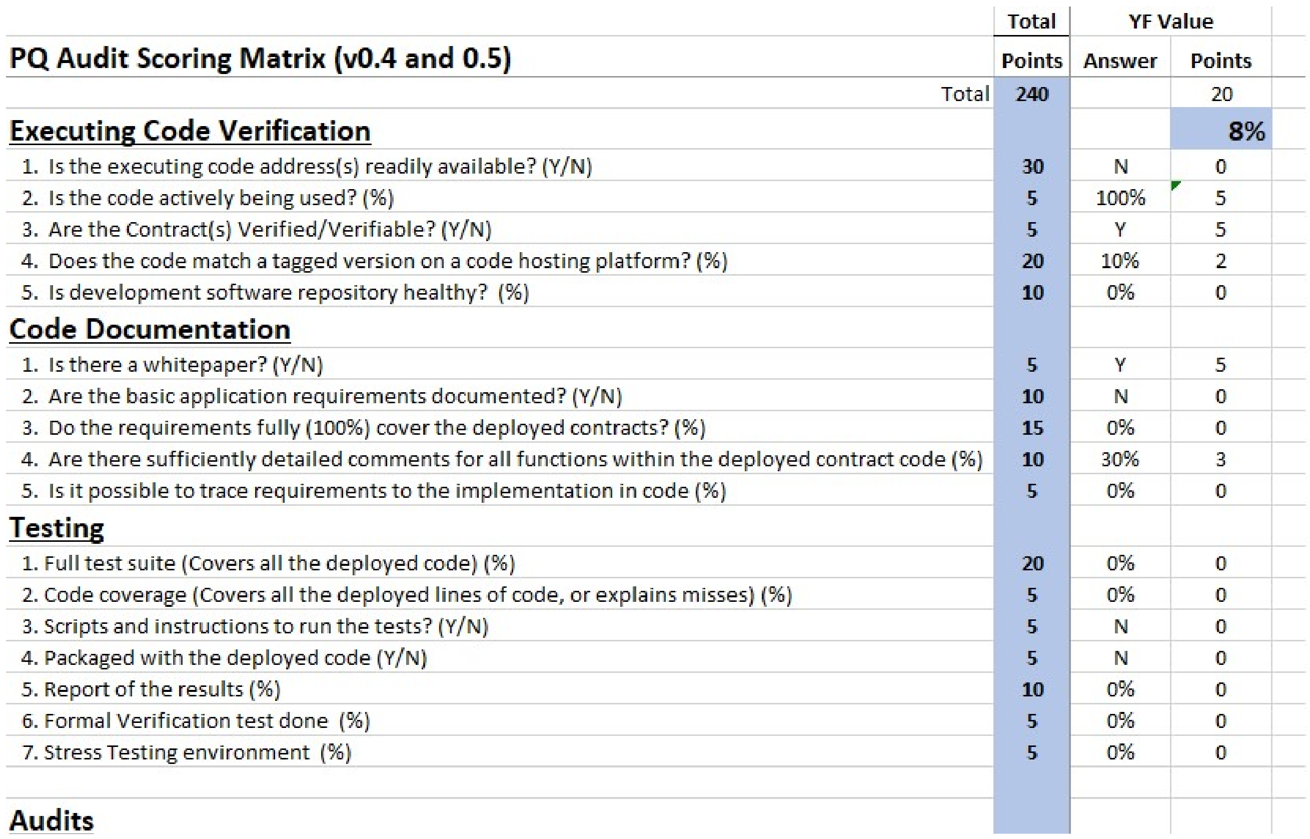
I started with Ethereum just before the DAO and that was a wonderful education. It showed the importance of code quality. The second Parity hack also showed the importance of good process. Here my aviation background offers some value. Aerospace knows how to make reliable code using quality processes.

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.

Career wise I am a business development manager for an avionics supplier.

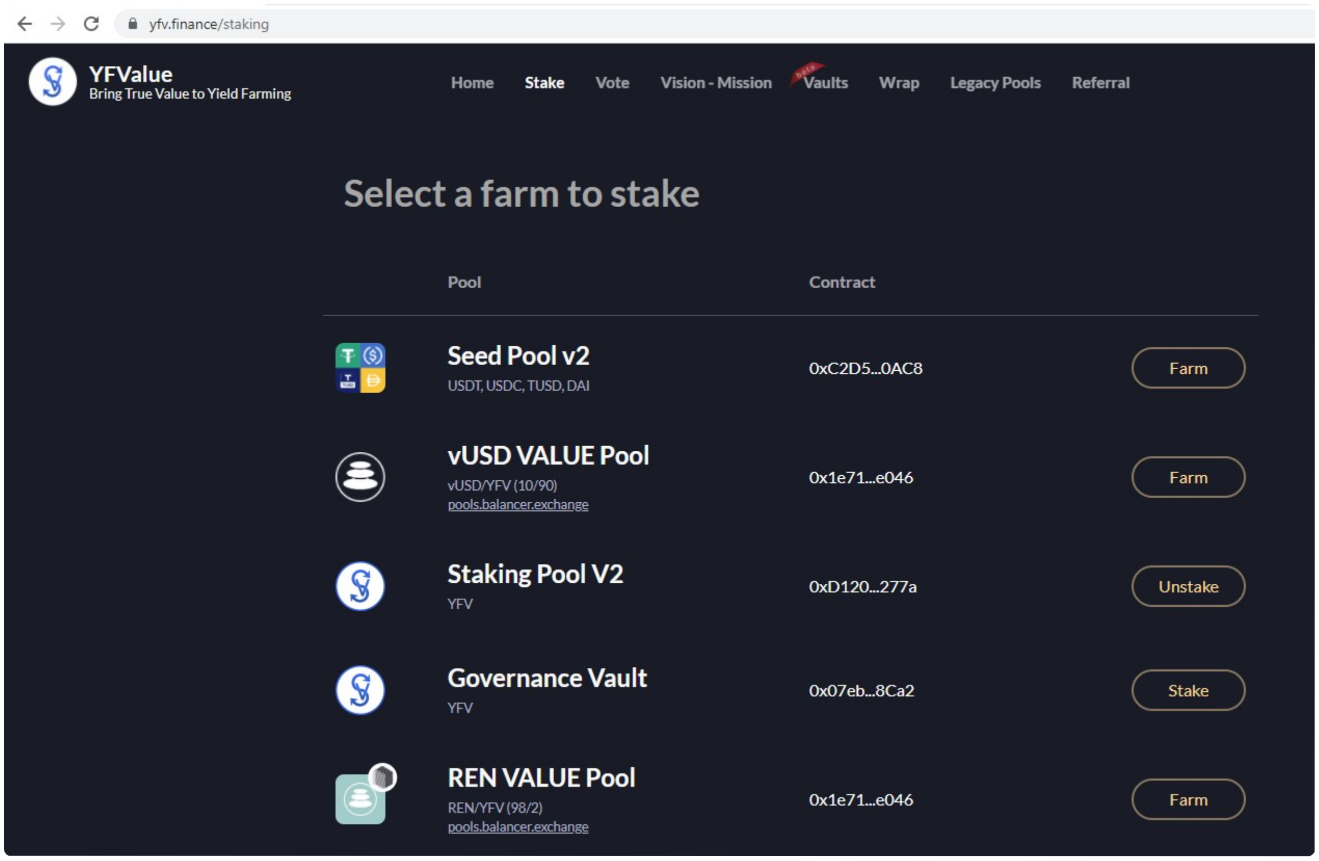
Scoring Appendix



	Total	YF Value	
PQ Audit Scoring Matrix (v0.4 and 0.5)	**Points**	**Answer**	**Points**
Total	**240**		20
Executing Code Verification			**8%**
1. Is the executing code address(s) readily available? (Y/N)	**30**	N	0
2. Is the code actively being used? (%)	**5**	100%	5
3. Are the Contract(s) Verified/Verifiable? (Y/N)	**5**	Y	5
4. Does the code match a tagged version on a code hosting platform? (%)	**20**	10%	2
5. Is development software repository healthy? (%)	**10**	0%	0
Code Documentation			
1. Is there a whitepaper? (Y/N)	**5**	Y	5
2. Are the basic application requirements documented? (Y/N)	**10**	N	0
3. Do the requirements fully (100%) cover the deployed contracts? (%)	**15**	0%	0
4. Are there sufficiently detailed comments for all functions within the deployed contract code (%)	**10**	30%	3
5. Is it possible to trace requirements to the implementation in code (%)	**5**	0%	0
Testing			
1. Full test suite (Covers all the deployed code) (%)	**20**	0%	0
2. Code coverage (Covers all the deployed lines of code, or explains misses) (%)	**5**	0%	0
3. Scripts and instructions to run the tests? (Y/N)	**5**	N	0
4. Packaged with the deployed code (Y/N)	**5**	N	0
5. Report of the results (%)	**10**	0%	0
6. Formal Verification test done (%)	**5**	0%	0
7. Stress Testing environment (%)	**5**	0%	0
Audits			

Audit done	70	0%	0
Section Scoring			
Executing Code Verification	70	17%	
Documentation	45	18%	
Testing	55	0%	
Audits	70	0%	

Executing Code Appendix



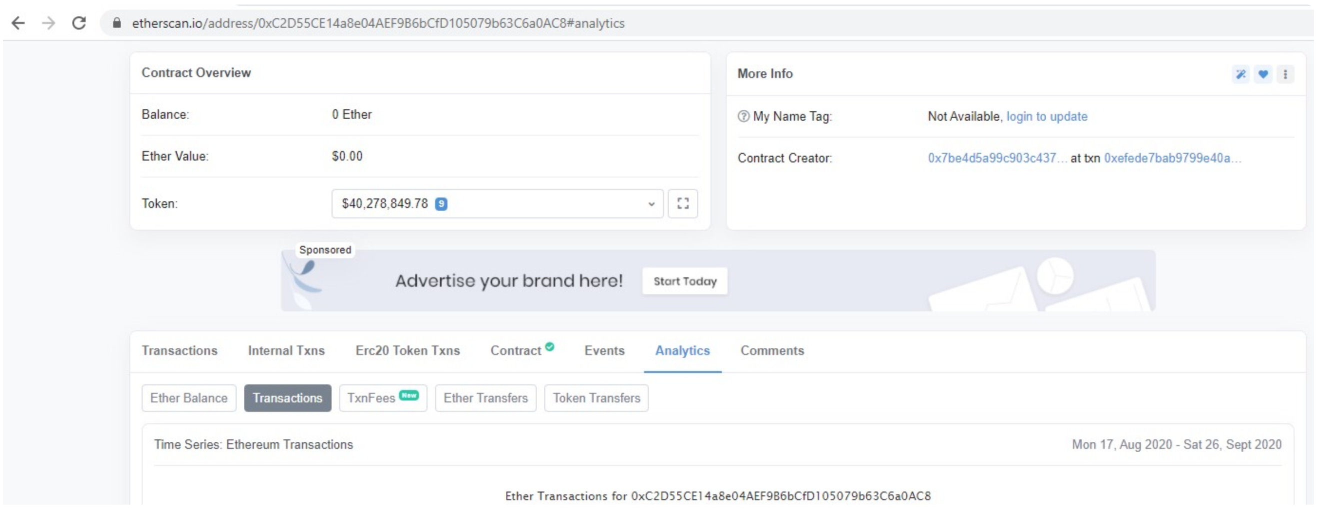
yfv.finance/staking

YFValue
Bring True Value to Yield Farming

Select a farm to stake

Pool	Contract	
Seed Pool v2 USDT, USDC, TUSD, DAI	0xC2D5...0AC8	Farm
vUSD VALUE Pool vUSD/YFV (10/90) pools.balancer.exchange	0x1e71...e046	Farm
Staking Pool V2 YFV	0xD120...277a	Unstake
Governance Vault YFV	0x07eb...8Ca2	Stake
REN VALUE Pool REN/YFV (98/2) pools.balancer.exchange	0x1e71...e046	Farm

Code Used Appendix



etherscan.io/address/0xC2D55CE14a8e04AEF9B6bCfD105079b63C6a0AC8#analytics

Contract Overview

Balance:	0 Ether
Ether Value:	$0.00
Token:	$40,278,849.78 9

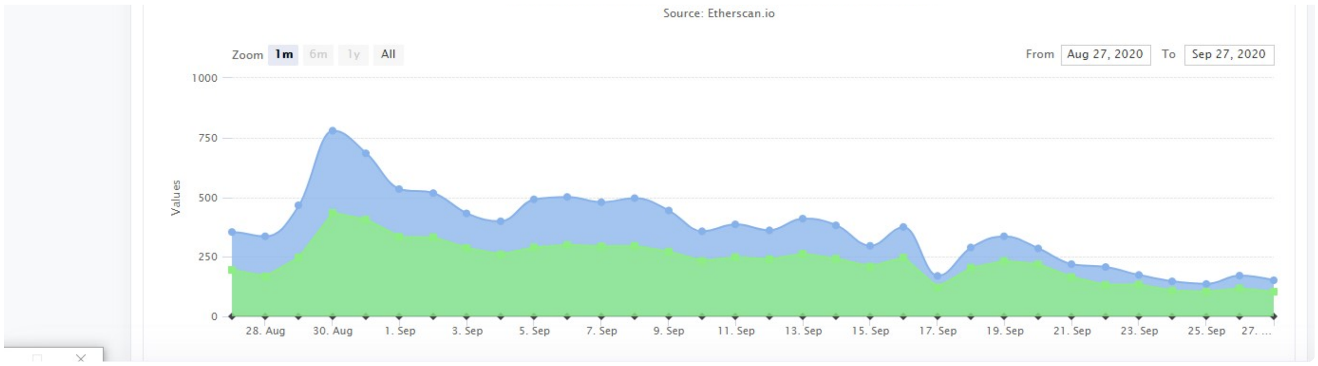
More Info

My Name Tag:	Not Available, login to update
Contract Creator:	0x7be4d5a99c903c437... at txn 0xefede7bab9799e40a...

Sponsored
Advertise your brand here! Start Today

Time Series: Ethereum Transactions
Mon 17, Aug 2020 - Sat 26, Sept 2020

Ether Transactions for 0xC2D55CE14a8e04AEF9B6bCfD105079b63C6a0AC8



Example Code Appendix

```
contract YFVStakeV2 is LPTokenWrapper, IRewardDistributionRecipient {
    IERC20 public vUSD = IERC20(0x1B8E12F839BD4e73A47adDF76cF7F0097d74c14C);
    IERC20 public vETH = IERC20(0x76A034e76Aa835363056dd418611E4f81870f16e);

    uint256 public vETH_REWARD_FRACTION_RATE = 1000;

    uint256 public constant DURATION = 7 days;
    uint8 public constant NUMBER_EPOCHS = 38;

    uint256 public constant REFERRAL_COMMISSION_PERCENT = 1;

    uint256 public currentEpochReward = 0;
    uint256 public totalAccumulatedReward = 0;
    uint8 public currentEpoch = 0;
    uint256 public starttime = 1598968800; // Tuesday, September 1, 2020 2:00:00 PM (GMT+0
    uint256 public periodFinish = 0;
    uint256 public rewardRate = 0;
    uint256 public lastUpdateTime;
    uint256 public rewardPerTokenStored;

    uint256 public constant DEFAULT_EPOCH_REWARD = 230000 * (10 ** 9); // 230,000 vUSD (an
    uint256 public constant TOTAL_REWARD = DEFAULT_EPOCH_REWARD * NUMBER_EPOCHS; // 8,740,

    uint256 public epochReward = DEFAULT_EPOCH_REWARD;
    uint256 public minStakingAmount = 90 ether;
    uint256 public unstakingFrozenTime = 40 hours;

    // ** DISABLED AT BEGINNING - WILL SET IT BY GOVERNANCE AFTER VIP-1.1
    // ** unlockWithdrawFee = 0.1%: stakers will need to pay 0.1% (sent to insurance fund)
    // ** lowStakeDepositFee = 0.1%: stakers still can stake with low amount but need to p
    //    specially, if lowStakeDepositFee = 10000 -> low amount stakers will not pay anyt
    // ** highStakeDepositFee = 0.1%: stakers need to pay 0.1% of extra amount more than 9
    uint256 public lowStakeDepositFee = 0; // per ten thousand (eg. 15 -> 0.15%)
    uint256 public highStakeDepositFee = 0; // per ten thousand (eg. 15 -> 0.15%)
    uint256 public unlockWithdrawFee = 0; // per ten thousand (eg. 15 -> 0.15%)

    address public yfvInsuranceFund = 0xb7b2Ea8A1198368f950834875047aA7294A2bDAa; // set t

    mapping(address => uint256) public userRewardPerTokenPaid;
    mapping(address => uint256) public rewards;
    mapping(address => uint256) public lastStakeTimes;

    mapping(address => uint256) public accumulatedStakingPower; // will accumulate every t

    mapping(address => bool) public whitelistedPools; // for stake on behalf

    event RewardAdded(uint256 reward);
    event YfvRewardAdded(uint256 reward);
    event Burned(uint256 reward);
    event Staked(address indexed user, uint256 amount, uint256 actualStakeAmount);
    event Withdrawn(address indexed user, uint256 amount, uint256 actualWithdrawAmount);
    event RewardPaid(address indexed user, uint256 reward);
    event CommissionPaid(address indexed user, uint256 reward);


```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complex
Solidity	8	2526	339	312	1875	358

Comments to Code 312 / 1875 = 17%