

# 0.7

## Abracadabra.money Process Quality Review

Score: 43%

### Overview

This is an [Abracadabra.Money](#) Process Quality Review completed on 13/10/2021. It was performed using the Process Review process (version 0.7.3) and is documented [here](#). The review was performed by Nick of DeFiSafety. Check out our [Telegram](#).

The final score of the review is **43%**, 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 blockchains used by this protocol. This report covers all of the blockchains upon which the protocol is deployed.

-  **Chain:** Ethereum, Avalanche, Binance Smart Chain, Fantom, Arbitrum

## 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 at website <https://docs.abracadabra.money/our-ecosystem/our-cauldrons-contract>, 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 more than 10 transactions a day on contract [Abracadabra.Money: yvstETH Market](#), 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/Abracadabra-money/magic-internet-money>

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%

At 79 commits and 11 branches, this repository's history is just shy of spellbinding but is certainly well on the way to becoming as enchanting as DeFi expects.

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: Yes

The founder of Abracadraba, Daniele Sesta, is public at <https://twitter.com/danielesesta>.

## 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.abracadabra.money/>

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

 **Answer:** No

Although dApp instructions are [identified](#), this is not considered as software function (code) 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%

While the dApp functions are identified, there is no explanation of code used in the deployed contracts. This requires greater elaboration on how the code performs the functions that the docs identify.

**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:** 28%

Code examples are in the [Appendix](#). As per the [SLOC](#), there is 29% 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%

Documentation identifies the contracts and what they do, but does not reference any code in the process.

#### 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:** 80%

Code examples are in the [Appendix](#). As per the [SLOC](#), there is 108% testing to code (TtC).

This score is guided by the Test to Code ratio (TtC). Generally a good test to code ratio is over 100%.

However the reviewers best judgement is the final deciding factor.

#### Guidance:

- 100% TtC > 120% Both unit and system test visible
- 80% TtC > 80% Both unit and system test visible
- 40% TtC < 80% Some tests visible
- 0% No tests obvious

How to improve this score:

This score can be 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:** 50%

No code coverage testing was found, but evidently some testing has occurred.

#### 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 be 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

Scripts were found at <https://github.com/Abracadabra-money/magic-internet-money/tree/main/scripts>.

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

 **Answer:** 0%

No report results were 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 formal verification was found.

#### 16) Stress Testing environment (%)

 **Answer:** 0%

No evidence of stress testing on a testnet was 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:** 20%

Abracadabra is unaudited. A changelog was provided to the DeFiSafety team, though this was not circulated to the public meaning we cannot factor this into the score.

#### Guidance:

- 100% Multiple Audits performed before deployment and results public and implemented or not required
- 90% Single audit performed before deployment and results public and implemented or not required
- 70% Audit(s) performed after deployment and no changes required. Audit report is public
- 50% Audit(s) performed after deployment and changes needed but not implemented
- 20% No audit performed
- 0% Audit Performed after deployment, existence is public, report is not public and no improvements deployed OR smart contract address' not found, (where question 1 is 0%)

Deduct 25% if code is in a private repo and no note from auditors that audit is applicable to deployed code

### 18) Is the bounty value acceptably high (%)

 **Answer:** 70%

Abracadabra offers an active program with a [top bounty of \\$100K](#).

#### 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: 40%

A mention that the protocol's governance decisions are subject to [team consideration](#) means that the protocol is controlled by the anonymous team. Additional governance info can be found in the [tokenomics](#) section of their docs, as well as in [this Medium article](#).

#### 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 (%)

ℹ Answer: 55%

a) All contracts are clearly labelled as upgradeable (or not) -- 15% -- certain protocol parameters are clearly labelled as upgradeable through voting in [this Medium article](#). However, the specific contracts are not explicitly mentioned.

b) The type of ownership is clearly indicated (OnlyOwner / MultiSig / Defined Roles) -- 30% -- admin ownership is [implied](#). In addition, MultiSig info can be found at

<https://docs.abracadabra.money/tokens/tokenomics>.

c) The capabilities for change in the contracts are described -- 10% -- [This Medium article](#) describes what parameters can be changed within the protocol, but doesn't touch upon the scope of these capabilities for change.

#### 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:** 30%

Although there is an acceptable amount of access control information, none of it relates directly to user investment safety, and it therefore more software-specific language.

### Guidance:

- 100% All the contracts are immutable
- 90% Description relates to investments safety and updates in clear, complete non-software 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%

No pause control documentation was found.

### 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	Abracadabra.money	
		Points	Answer	Points
<b>PQ Audit Scoring Matrix (v0.7)</b>		Total		
		260		111.4
				<b>43%</b>
<b>Code and Team</b>				
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	y	15
<b>Code Documentation</b>				
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	28%	1.4
10) Is it possible to trace from software documentation to the implementation in code (%)		10	0%	0
<b>Testing</b>				
11) Full test suite (Covers all the deployed code) (%)		20	80%	16
12) Code coverage (Covers all the deployed lines of code, or explains misses) (%)		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
<b>Security</b>				
17) Did 3rd Party audits take place? (%)		70	20%	14
18) Is the bug bounty acceptable high? (%)		10	70%	7
<b>Access Controls</b>				
19) Can a user clearly and quickly find the status of the admin controls		5	40%	2
20) Is the information clear and complete		10	55%	5.5
21) Is the information in non-technical terms		10	30%	3
22) Is there Pause Control documentation including records of tests		10	0%	0
<b>Section Scoring</b>				
Code and Team		50	100%	
Documentation		45	14%	
Testing		50	47%	
Security		80	26%	
Access Controls		35	30%	

## Executing Code Appendix

### Ethereum Mainnet Markets:

Active Markets:

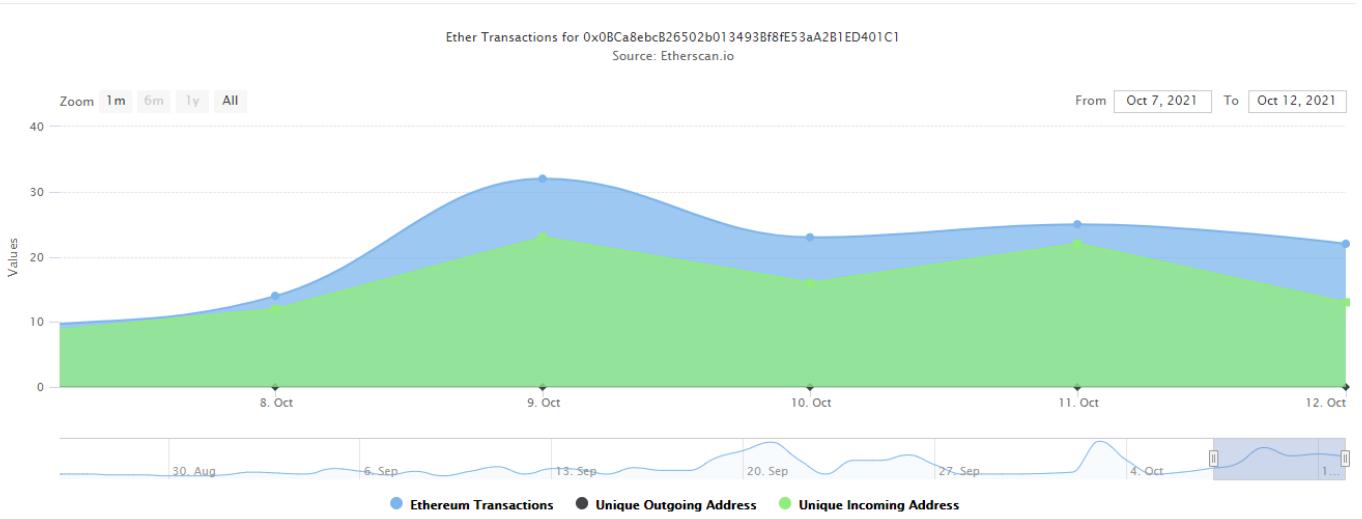
AI CX: 0x7b7473a76D6ae86CE19f7352A1E89F6C9dc39020

```

Cvx3pool: 0x806e16ec797c69afa8590A55723CE4CC1b54050E
FTM: 0x05500e2Ee779329698DF35760bEdcAAC046e7C27
wsOHM: 0x003d5A75d284824Af736df51933be522DE9Eed0f
xSUSHI: 0x98a84EfF6e008c5ed0289655CcdCa899bcb6B99F
yvcrvIB: 0xEBfDe87310dc22404d918058FAa4D56DC4E93f0A
yvstETH: 0x0BCa8ebcB26502b013493Bf8fE53aA2B1ED401C1
yvUSDC v2: 0x6cbAFEE1FaB76cA5B5e144c43B3B50d42b7C8c8f
yvUSDT v2: 0x551a7CfF4de931F32893c928bBc3D25bF1Fc5147
yvWETH v2: 0x920D9BD936Da4eAFb5E25c6bDC9f6CB528953F9f
yvYFI: 0xFFbF4892822e0d552CFF317F65e1eE7b5D3d9aE6
sSPELL: 0xC319EEa1e792577C319723b5e60a15dA3857E7da
cvxtricrypto2: 0x4EAeD76C3A388f4a841E9c765560BBe7B3E4B3A0

```

## Code Used Appendix



## Example Code Appendix

```

1 contract sSpell is IERC20, Domain {
2     using BoringMath for uint256;
3     using BoringMath128 for uint128;
4     using BoringERC20 for IERC20;
5
6     string public constant symbol = "sSPELL";
7     string public constant name = "Staked Spell Tokens";
8     uint8 public constant decimals = 18;
9     uint256 public override totalSupply;
10    uint256 private constant LOCK_TIME = 24 hours;
11
12    IERC20 public immutable token;
13
14    constructor(IERC20 _token) public {
15        token = _token;
16    }
17

```

```

18     struct User {
19         uint128 balance;
20         uint128 lockedUntil;
21     }
22
23     /// @notice owner > balance mapping.
24     mapping(address => User) public users;
25     /// @notice owner > spender > allowance mapping.
26     mapping(address => mapping(address => uint256)) public override allowance;
27     /// @notice owner > nonce mapping. Used in `permit`.
28     mapping(address => uint256) public nonces;
29
30     event Transfer(address indexed _from, address indexed _to, uint256 _value);
31     event Approval(address indexed _owner, address indexed _spender, uint256 _value);
32
33     function balanceOf(address user) public view override returns (uint256 balance) {
34         return users[user].balance;
35     }
36
37     function _transfer(
38         address from,
39         address to,
40         uint256 shares
41    ) internal {
42         User memory fromUser = users[from];
43         require(block.timestamp >= fromUser.lockedUntil, "Locked");
44         if (shares != 0) {
45             require(fromUser.balance >= shares, "Low balance");
46             if (from != to) {
47                 require(to != address(0), "Zero address"); // Moved down so other failed cases
48                 User memory toUser = users[to];
49                 users[from].balance = fromUser.balance - shares.to128(); // Underflow is checked
50                 users[to].balance = toUser.balance + shares.to128(); // Can't overflow because
51             }
52         }
53         emit Transfer(from, to, shares);
54     }
55
56     function _useAllowance(address from, uint256 shares) internal {
57         if (msg.sender == from) {
58             return;
59         }
60         uint256 spenderAllowance = allowance[from][msg.sender];
61         // If allowance is infinite, don't decrease it to save on gas (breaks with EIP-20)
62         if (spenderAllowance != type(uint256).max) {
63             require(spenderAllowance >= shares, "Low allowance");
64             allowance[from][msg.sender] = spenderAllowance - shares; // Underflow is checked
65         }
66     }
67
68     /// @notice Transfers `shares` tokens from `msg.sender` to `to`.
69     /// @param to The address to move the tokens.

```

```

70     /// @param shares of the tokens to move.
71     /// @return (bool) Returns True if succeeded.
72     function transfer(address to, uint256 shares) public returns (bool) {
73         _transfer(msg.sender, to, shares);
74         return true;
75     }
76
77     /// @notice Transfers `shares` tokens from `from` to `to`. Caller needs approval for `shares`.
78     /// @param from Address to draw tokens from.
79     /// @param to The address to move the tokens.
80     /// @param shares The token shares to move.
81     /// @return (bool) Returns True if succeeded.
82     function transferFrom(
83         address from,
84         address to,
85         uint256 shares
86     ) public returns (bool) {
87         _useAllowance(from, shares);
88         _transfer(from, to, shares);
89         return true;
90     }
91
92     /// @notice Approves `amount` from sender to be spend by `spender`.
93     /// @param spender Address of the party that can draw from msg.sender's account.
94     /// @param amount The maximum collective amount that `spender` can draw.
95     /// @return (bool) Returns True if approved.
96     function approve(address spender, uint256 amount) public override returns (bool) {
97         allowance[msg.sender][spender] = amount;
98         emit Approval(msg.sender, spender, amount);
99         return true;
100    }
101
102    // solhint-disable-next-line func-name-mixedcase
103    function DOMAIN_SEPARATOR() external view returns (bytes32) {
104        return _domainSeparator();
105    }
106
107    // keccak256("Permit(address owner,address spender,uint256 value,uint256 nonce,uint256
108    bytes32 private constant PERMIT_SIGNATURE_HASH = 0x6e71edae12b1b97f4d1f60370fef10105fa;
109
110    /// @notice Approves `value` from `owner_` to be spend by `spender`.
111    /// @param owner_ Address of the owner.
112    /// @param spender The address of the spender that gets approved to draw from `owner_`.
113    /// @param value The maximum collective amount that `spender` can draw.
114    /// @param deadline This permit must be redeemed before this deadline (UTC timestamp in seconds).
115    function permit(
116        address owner_,
117        address spender,
118        uint256 value,
119        uint256 deadline,
120        uint8 v,
121        bytes32 r,
122        bytes32 s

```

```

123     ) external override {
124         require(owner_ != address(0), "Zero owner");
125         require(block.timestamp < deadline, "Expired");
126         require(
127             ecrecover(_getDigest(keccak256(abi.encode(PERMIT_SIGNATURE_HASH, owner_, spender,
128                         owner_),
129                         "Invalid Sig"
130                     ));
131             allowance[owner_][spender] = value;
132             emit Approval(owner_, spender, value);
133         }
134
135     /// math is ok, because amount, totalSupply and shares is always 0 <= amount <= 100.000
136     /// theoretically you can grow the amount/share ratio, but it's not practical and useless
137     function mint(uint256 amount) public returns (bool) {
138         require(msg.sender != address(0), "Zero address");
139         User memory user = users[msg.sender];
140
141         uint256 totalTokens = token.balanceOf(address(this));
142         uint256 shares = totalSupply == 0 ? amount : (amount * totalSupply) / totalTokens;
143         user.balance += shares.to128();
144         user.lockedUntil = (block.timestamp + LOCK_TIME).to128();
145         users[msg.sender] = user;
146         totalSupply += shares;
147
148         token.safeTransferFrom(msg.sender, address(this), amount);
149
150         emit Transfer(address(0), msg.sender, shares);
151         return true;
152     }
153
154     function _burn(
155         address from,
156         address to,
157         uint256 shares
158     ) internal {
159         require(to != address(0), "Zero address");
160         User memory user = users[from];
161         require(block.timestamp >= user.lockedUntil, "Locked");
162         uint256 amount = (shares * token.balanceOf(address(this))) / totalSupply;
163         users[from].balance = user.balance.sub(shares.to128()); // Must check underflow
164         totalSupply -= shares;
165
166         token.safeTransfer(to, amount);
167
168         emit Transfer(from, address(0), shares);
169     }
170
171     function burn(address to, uint256 shares) public returns (bool) {
172         _burn(msg.sender, to, shares);
173         return true;
174     }
175

```

```

176     function burnFrom(
177         address from,
178         address to,
179         uint256 shares
180     ) public returns (bool) {
181         _useAllowance(from, shares);
182         _burn(from, to, shares);
183         return true;

```

## SLOC Appendix

### Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complexity
Solidity	14	2910	415	554	1941	387

Comments to Code  $554/1941 = 29\%$

### Javascript Tests

Language	Files	Lines	Blanks	Comments	Code	Complexity
JavaScript	13	2594	391	107	2096	61

Tests to Code  $2096/1941 = 108\%$