MakerDAO PQ Review

Score: 84%

Overview

This is a Marker DAO Process Quality Review completed on 4 May 2021. It was performed using the Process Review process (version 0.7) and is documented here. The review was performed by Rex of DeFiSafety. Check out our Telegram.

The final score of the review is 84%, a strong pass. 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

Chain

This section indicates the blockchain used by this protocol. The chains and their multiples are explained in this document.

✓ **Chain: Ethereum**

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%

You have to dig a bit to find the deployed contract addresses. First pick Developers from the website, then Documentation. From there when you dig to a particular module, with the detailed documentation they have the Etherscan address. A bit different, but effective. 3 clicks to an address, so 100%. See Appendix for an example,

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

They are available at website ____ as indicated in the Appendix.

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 addresses is up to date. This is a very important question wrt to the final score.

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

> ✓ Answer: 100%

Activity is over 2,000 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

3) Is there a public software repository? (Y/N)

> ✓ Answer: Yes

GitHub: https://github.com/makerdao/dss

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: 100%

With 396 commits and 18 branches, this is a healthy repo

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 10 commits

5) Is the team public (not anonymous)? (Y/N)

> ✓ Answer: Yes

Yes, team: https://makerdao.com/en/about

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://makerdao.com/en/whitepaper/

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

> ✓ Answer: Yes

Location: https://docs.makerdao.com/maker-protocol-101

8) Does the software function documentation fully (100%) cover the deployed contracts? (%)

> ✓ Answer: 100%

In the Maker Docs there is excellent documentation on all contracts in the protocol.

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: 25

As per the code examples there is only minimal commenting in the actual code. Despite excellent top level documentation, there is little connection and the code alone does not explain or connect with the documentation.

Code examples are in the Appendix. As per the SLOC, there is 21% 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: 60%

The detailed docs give clear, but not explicit traceability with examples within the docs.

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: 100%

There is a separate test file for almost each solidity file. The test files stay with each branch of the source files. There is a 171% of tests to code as per the SLOC data

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: 70%

There is no evidence of organized collection of code coverage. The CodeCov in the GitHub is enabled, but empty. However with the comprehensive set of tests, some coverage would be found. 50% as per guidance below.

However on page 44 of the ToBits audit is the code coverage. There is no summary but the main files have high coverage so for this we will score 70%.

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: Yes

Yes in travis.yml.

14) Report of the results (%)

ⓘ Answer: 70%

However on page 44 of the ToBits audit is the code coverage. There is no summary but the main files have high coverage so for this we will score 70%, with other details of a test results.

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: 100%

A report on the Formal Verification done: https://security.makerdao.com/formal-verification

16) Stress Testing environment (%)

✓ Answer: 100%

A report on the Runtime verification done: https://forum.makerdao.com/t/publication-of-the-runtime-verification-audit/976

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%

Two detailed and comprehensive audits took place on the MakerDao contracts before deployment. Some vulnerabilities were found. We assume these were fixed, but actual evidence of the fixes were not seen. Overall an excellent securities effort.

Trail of Bits Audit: https://github.com/makerdao/mcd-security/blob/master/Audit%20Reports/TOB_MakerDAO_Final_Report.pdf

Peckshield Audit: https://github.com/makerdao/mcd-security/blob/master/Audit%20Reports/PeckShield_Final_Audit_Report.pdf

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%)

18) Is the bounty value acceptably high (%)

✓ Answer: 70%

Bug Bounty Location:

Guidance:

100%	Bounty is 10% TVL or at least 1M
90%	Bounty is 5% TVL or at least 500k
70%	Bounty is 100k or over
40%	Bounty is 50k or over
20%	Bug bounty program bounty is less than 50k
0%	No bug bounty program offered

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 admin controls (%)

> Answer: 20%

The location of the Pause capability is clearly indicated, but no text on the ability to update contracts is found.

Location: https://docs.makerdao.com/smart-contract-modules/shutdown

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%

Guidance:
All the contracts are immutable -- 100% OR

All contracts are clearly labelled as upgradeable (or not) -- 30% AND
The type of ownership is clearly indicated (OnlyOwner / MultiSig / Defined Roles) -- 30% AND
The capabilities for change in the contracts are described -- 30%

21) Is the information in non-technical terms that pertain to the investments (%)

⚠ Answer: 0%

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

22) Is there Pause Control documentation including records of tests (%)

✓ Answer: 80%

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

The contracts that are updateable clearly indicated

Appendices

Author Details

The author of this review is Rex of DeFi Safety.

Email : rex@defisafety.com Twitter : @defisafety

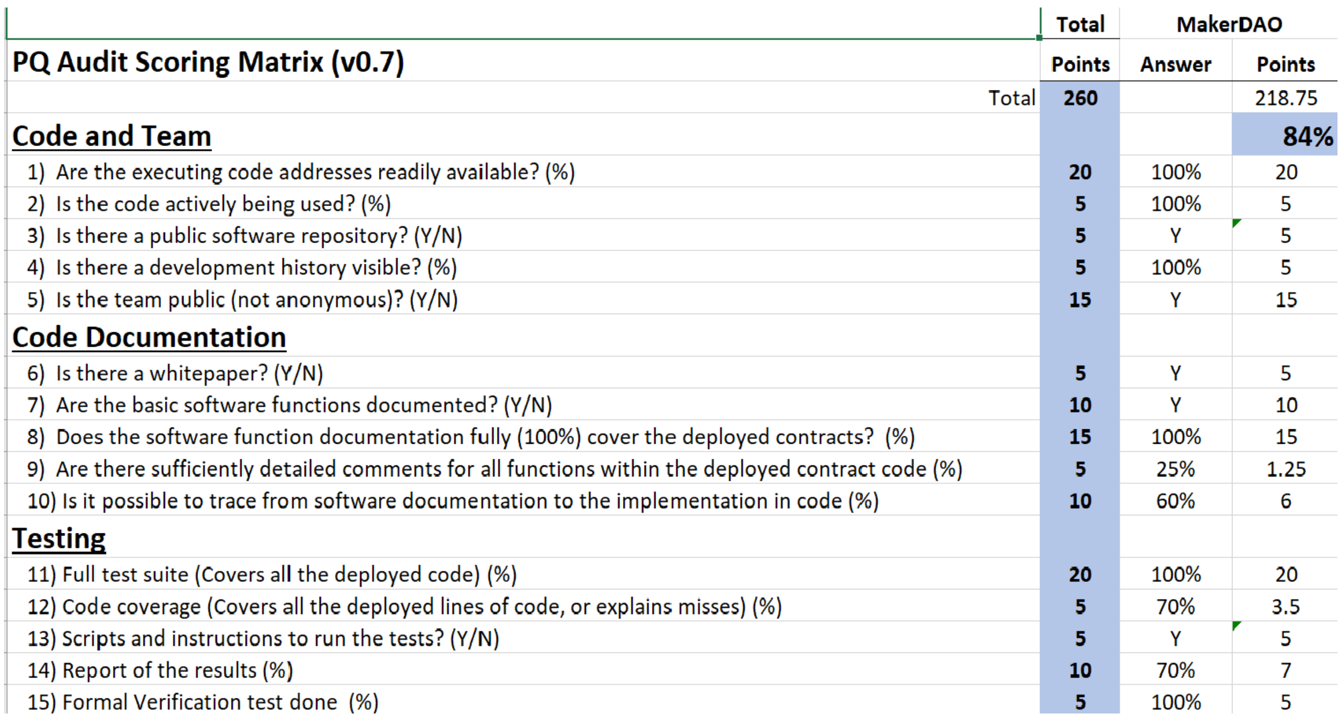
I started with Ethereum just before the DAO and that was a wonderful education. It showed the importance of code quality. The second Parity hack also showed the importance of good process. Here my aviation background offers some value. Aerospace knows how to make reliable code using quality processes.

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.

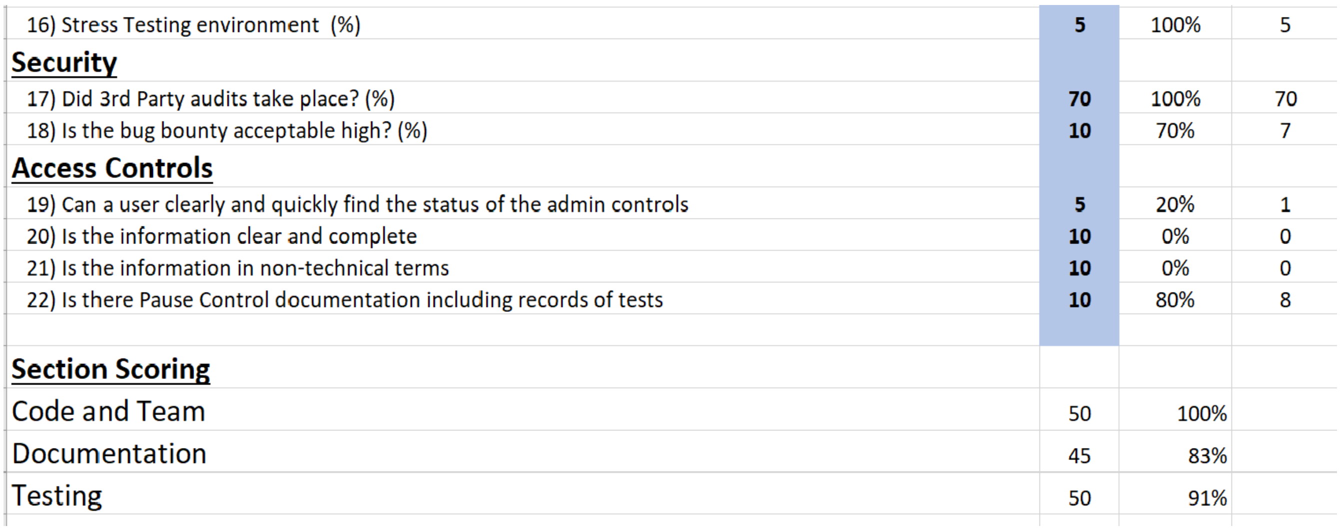
Process Quality Reviews are an extension of the SecurEth guidelines that will further increase the quality processes in Solidity and Vyper development.

DeFiSafety is my full time gig and we are working on funding vehicles for a permanent staff.

Scoring Appendix

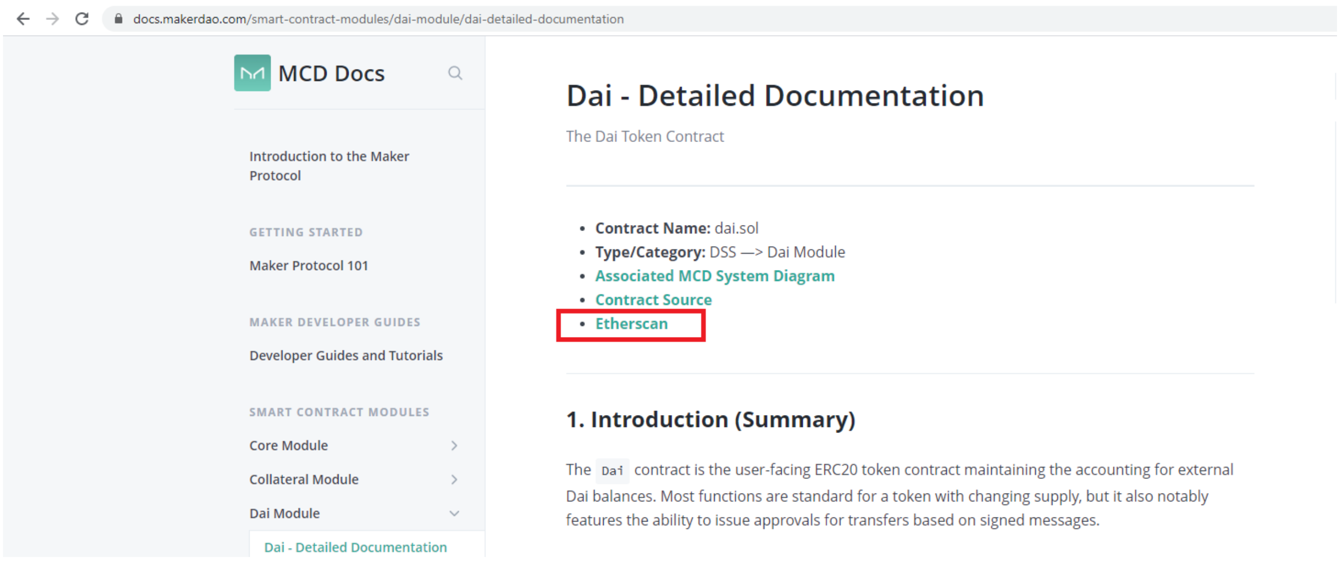


	Total	MakerDAO	
PQ Audit Scoring Matrix (v0.7)	**Points**	**Answer**	**Points**
Total	**260**		218.75
Code and Team			**84%**
1) Are the executing code addresses readily available? (%)	**20**	100%	20
2) Is the code actively being used? (%)	**5**	100%	5
3) Is there a public software repository? (Y/N)	**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
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%) cover the deployed contracts? (%)	**15**	100%	15
9) Are there sufficiently detailed comments for all functions within the deployed contract code (%)	**5**	25%	1.25
10) Is it possible to trace from software documentation to the implementation in code (%)	**10**	60%	6
Testing			
11) Full test suite (Covers all the deployed code) (%)	**20**	100%	20
12) Code coverage (Covers all the deployed lines of code, or explains misses) (%)	**5**	70%	3.5
13) Scripts and instructions to run the tests? (Y/N)	**5**	Y	5
14) Report of the results (%)	**10**	70%	7
15) Formal Verification test done (%)	**5**	100%	5



16) Stress Testing environment (%)	**5**	100%	5
Security			
17) Did 3rd Party audits take place? (%)	**70**	100%	70
18) Is the bug bounty acceptable high? (%)	**10**	70%	7
Access Controls			
19) Can a user clearly and quickly find the status of the admin controls	**5**	20%	1
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**	80%	8
Section Scoring			
Code and Team	50	100%	
Documentation	45	83%	
Testing	50	91%	

Executing Code Appendix



docs.makerdao.com/smart-contract-modules/dai-module/dai-detailed-documentation

MCD Docs

Introduction to the Maker Protocol

GETTING STARTED

Maker Protocol 101

MAKER DEVELOPER GUIDES

Developer Guides and Tutorials

SMART CONTRACT MODULES

Core Module

Collateral Module

Dai Module

Dai - Detailed Documentation

Dai - Detailed Documentation

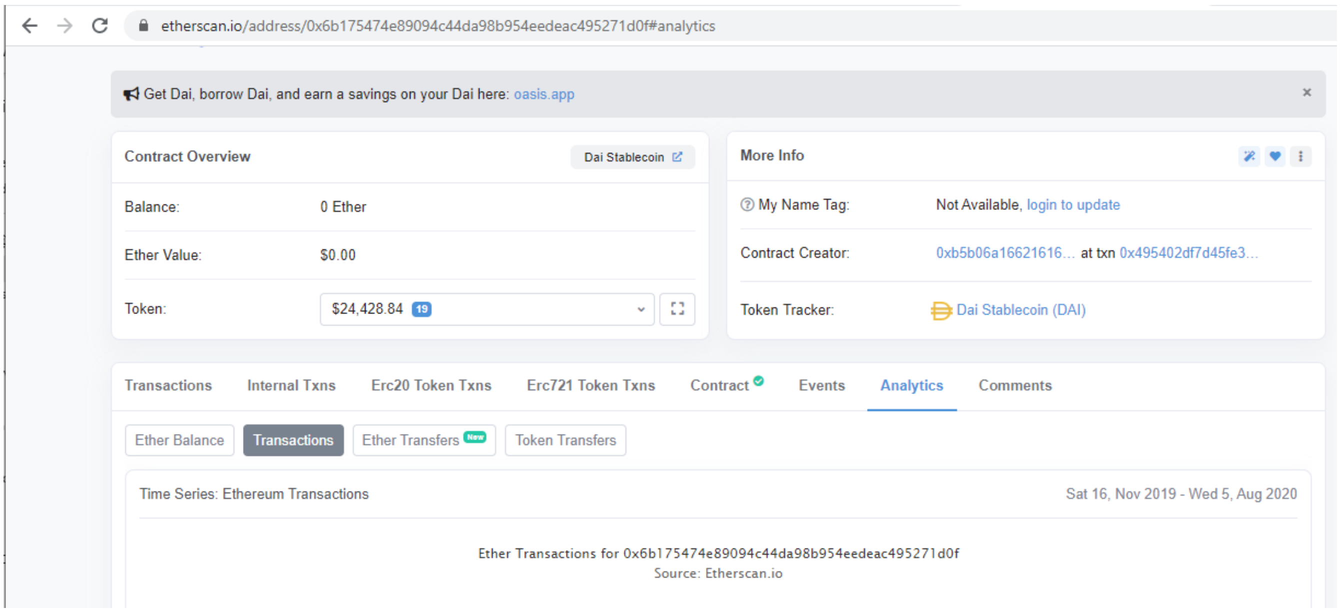
The Dai Token Contract

- **Contract Name:** dai.sol
- **Type/Category:** DSS —> Dai Module
- **Associated MCD System Diagram**
- **Contract Source**
- **Etherscan**

1. Introduction (Summary)

The `Dai` contract is the user-facing ERC20 token contract maintaining the accounting for external Dai balances. Most functions are standard for a token with changing supply, but it also notably features the ability to issue approvals for transfers based on signed messages.

Code Used Appendix



etherscan.io/address/0x6b175474e89094c44da98b954eedeac495271d0f#analytics

Get Dai, borrow Dai, and earn a savings on your Dai here: oasis.app

Contract Overview	Dai Stablecoin
Balance:	0 Ether
Ether Value:	$0.00
Token:	$24,428.84 19

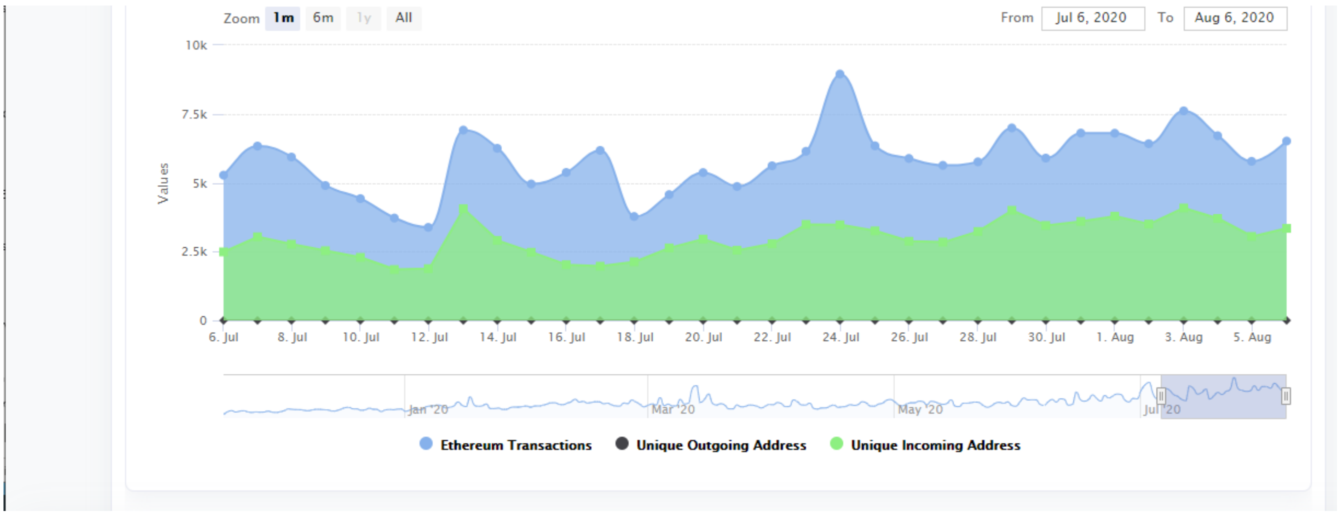
More Info	
My Name Tag:	Not Available, login to update
Contract Creator:	0xb5b06a16621616... at txn 0x495402df7d45fe3...
Token Tracker:	Dai Stablecoin (DAI)

Transactions | Internal Txns | Erc20 Token Txns | Erc721 Token Txns | Contract | Events | Analytics | Comments

Ether Balance | Transactions | Ether Transfers New | Token Transfers

Time Series: Ethereum Transactions Sat 16, Nov 2019 - Wed 5, Aug 2020

Ether Transactions for 0x6b175474e89094c44da98b954eedeac495271d0f
Source: Etherscan.io



Example Code Appendix

```
contract Vat {
    // --- Auth ---
    mapping (address => uint) public wards;
    function rely(address usr) external note auth { require(live == 1, "V
    function deny(address usr) external note auth { require(live == 1, "V
    modifier auth {
        require(wards[msg.sender] == 1, "Vat/not-authorized");
        _;
    }

    mapping(address => mapping (address => uint)) public can;
    function hope(address usr) external note { can[msg.sender][usr] = 1;
    function nope(address usr) external note { can[msg.sender][usr] = 0;
    function wish(address bit, address usr) internal view returns (bool)
        return either(bit == usr, can[bit][usr] == 1);
    }

    // --- Data ---
    struct Ilk {
        uint256 Art;   // Total Normalised Debt     [wad]
        uint256 rate;  // Accumulated Rates         [ray]
        uint256 spot;  // Price with Safety Margin  [ray]
        uint256 line;  // Debt Ceiling              [rad]
        uint256 dust;  // Urn Debt Floor            [rad]
    }
    struct Urn {
        uint256 ink;   // Locked Collateral  [wad]
        uint256 art;   // Normalised Debt    [wad]
    }

    mapping (bytes32 => Ilk)                       public ilks;
    mapping (bytes32 => mapping (address => Urn )) public urns;
    mapping (bytes32 => mapping (address => uint)) public gem;  // [wad]
    mapping (address => uint256)                   public dai;  // [rad]
    mapping (address => uint256)                   public sin;  // [rad]

    uint256 public debt;  // Total Dai Issued    [rad]
    uint256 public vice;  // Total Unbacked Dai  [rad]
    uint256 public Line;  // Total Debt Ceiling  [rad]
    uint256 public live;  // Access Flag

    // --- Logs ---
    event LogNote(
        bytes4   indexed  sig,
        bytes32  indexed  arg1,
        bytes32  indexed  arg2,
        bytes32  indexed  arg3,
        bytes             data
    ) anonymous;

    modifier note {
        _;
        assembly {
            // log an 'anonymous' event with a constant 6 words of callda
            // and four indexed topics: the selector and the first three a
            let mark := msize                         // end of memory en
            mstore(0x40, add(mark, 288))              // update free memo
            mstore(mark, 0x20)                        // bytes type data 
            mstore(add(mark, 0x20), 224)              // bytes size (padd
            calldatacopy(add(mark, 0x40), 0, 224)     // bytes payload
            log4(mark, 288,                           // calldata
                 shl(224, shr(224, calldataload(0))), // msg.sig
                 calldataload(4),                     // arg1
                 calldataload(36),                    // arg2
                 calldataload(68)                     // arg3
                )
        }
    }

    // --- Init ---
    constructor() public {
        wards[msg.sender] = 1;
        live = 1;
    }

    // --- Math ---
    function add(uint x, int y) internal pure returns (uint z) {
        z = x + uint(y);
        require(y >= 0 || z <= x);
        require(y <= 0 || z >= x);
    }
    function sub(uint x, int y) internal pure returns (uint z) {
        z = x - uint(y);
        require(y <= 0 || z <= x);
        require(y >= 0 || z >= x);
    }
    function mul(uint x, int y) internal pure returns (int z) {
        z = int(x) * y;
        require(int(x) >= 0);
        require(y == 0 || z / y == int(x));
    }
    function add(uint x, uint y) internal pure returns (uint z) {
        require((z = x + y) >= x);
    }
    function sub(uint x, uint y) internal pure returns (uint z) {
        require((z = x - y) <= x);
    }
    function mul(uint x, uint y) internal pure returns (uint z) {
        require(y == 0 || (z = x * y) / y == x);
    }

    // --- Administration ---
    function init(bytes32 ilk) external note auth {
        require(ilks[ilk].rate == 0, "Vat/ilk-already-init");
        ilks[ilk].rate = 10 ** 27;
    }
    function file(bytes32 what, uint data) external note auth {
        require(live == 1, "Vat/not-live");
        if (what == "Line") Line = data;
        else revert("Vat/file-unrecognized-param");
    }
    function file(bytes32 ilk, bytes32 what, uint data) external note auth
        require(live == 1, "Vat/not-live");
        if (what == "spot") ilks[ilk].spot = data;
        else if (what == "line") ilks[ilk].line = data;
        else if (what == "dust") ilks[ilk].dust = data;
        else revert("Vat/file-unrecognized-param");
    }
    function cage() external note auth {
        live = 0;
    }

```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complexity
Solidity	13	2155	255	463	1437	301

Comments to Code: 463/ 2155 = 21%

Tests

Language	Files	Lines	Blanks	Comments	Code	Complexity
Solidity	10	3163	460	245	2458	15

Tests to Code 2458/ 1437= 171%