Akropolis Dex Process Quality Review

Score 86%

This is a Process Quality Audit completed on 4 September 2020. It was performed using the Process Audit process (version 0.5) and is documented here. The audit was performed by ShinkaRex of Caliburn Consulting. Check out our Telegram.

The final score of the audit is 86%, a great score. The breakdown of the scoring is in Scoring Appendix.

Summary of the Process

Very simply, the audit 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

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 audited and its corresponding software repository. The document explaining these questions is here. This audit 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: Yes

They are available at Address 0xc88F54A79CaE4C125D7A8c2Cf811daaE78b07D64 which proxies to 0xEca1B5B114146688E3f10b270c42336bEA8A84C7 as indicated in the Appendix. This Audit only covers the contract *FundsModule.sol*.

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: 70%

Activity is about 3 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: Y

0xEca1B5B114146688E3f10b270c42336bEA8A84C7 is the Etherscan verified contract address.

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

✓ Answer: 100%

A clearly labelled mainnet release, clear list of mainnet addresses and a perfect match. Does not get easier than that (once you bounce over the proxy). Thank-you Spartans ;)

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/akropolisio/sparta

Deployed contracts in the following file;

Sparta_Deployed.rar Sparta_Deployed.rar - 29KB

Matching Repository: https://github.com/akropolisio/sparta/releases/tag/v1.0

Is development software repository healthy? (%)

Answer: 100%

36 branches, 561 commits, this is where Akropolis have worked for some time.

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

Location: https://wiki.akropolis.io/wp2019/

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

> ✓ Answer: Y

Location: https://wiki.akropolis.io/sparta/ and https://github.com/akropolisio/sparta readme

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

> ✓ Answer: 80%

With the readme in the GitHub, all the contracts are described well. In addition the perspective of the deployment plan and exact contract calls for each function is very useful. the docs do not get down to the function level, meaning this does not get full score.

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: 60%

NATSpec formats in the comments, consistently before each function.

Code examples are in the Appendix. As per the SLOC, there is 38% 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: 20%

Traceability is very limited. The readme gives quite a bit of information but it is not very connected to the 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;

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: 100%

The test suite is in Typescript. There is a 146% test to code ratio

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: 82%

Coverage badge on GitHub has 82% value based on GitHub "coveralls".

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

I could not see any instructions on scripts, running 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.

Packaged with the deployed code (Y/N)

✓ Answer: Y

Report of the results (%)

! Answer: 50%

The "coveralls" function does give a good report on the areas not covered by the code coverage. What is missing is explanation. For instance, the interfaces directory has no coverage at all. There may be a very good reason for this, which would increase the ballot coverage significantly. However, no explanations on the "misses" on coverage are available.

We give 50% as the coveralls report is far more than we see on most repositories.

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 evidence of formal verification is seen.

Stress Testing environment (%)

⚠ Answer: 0%

While I'm sure there was a testnet used during development, I can see no evidence of it.

Audits

✓ Answer: 100%

Two audits, with corrections are included. The audits are referenced at https://github.com/akropolisio/sparta/tree/develop/audit.

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 audit is Rex of Caliburn Consulting.

Email : rex@defisafety.com Twitter : @defisafety

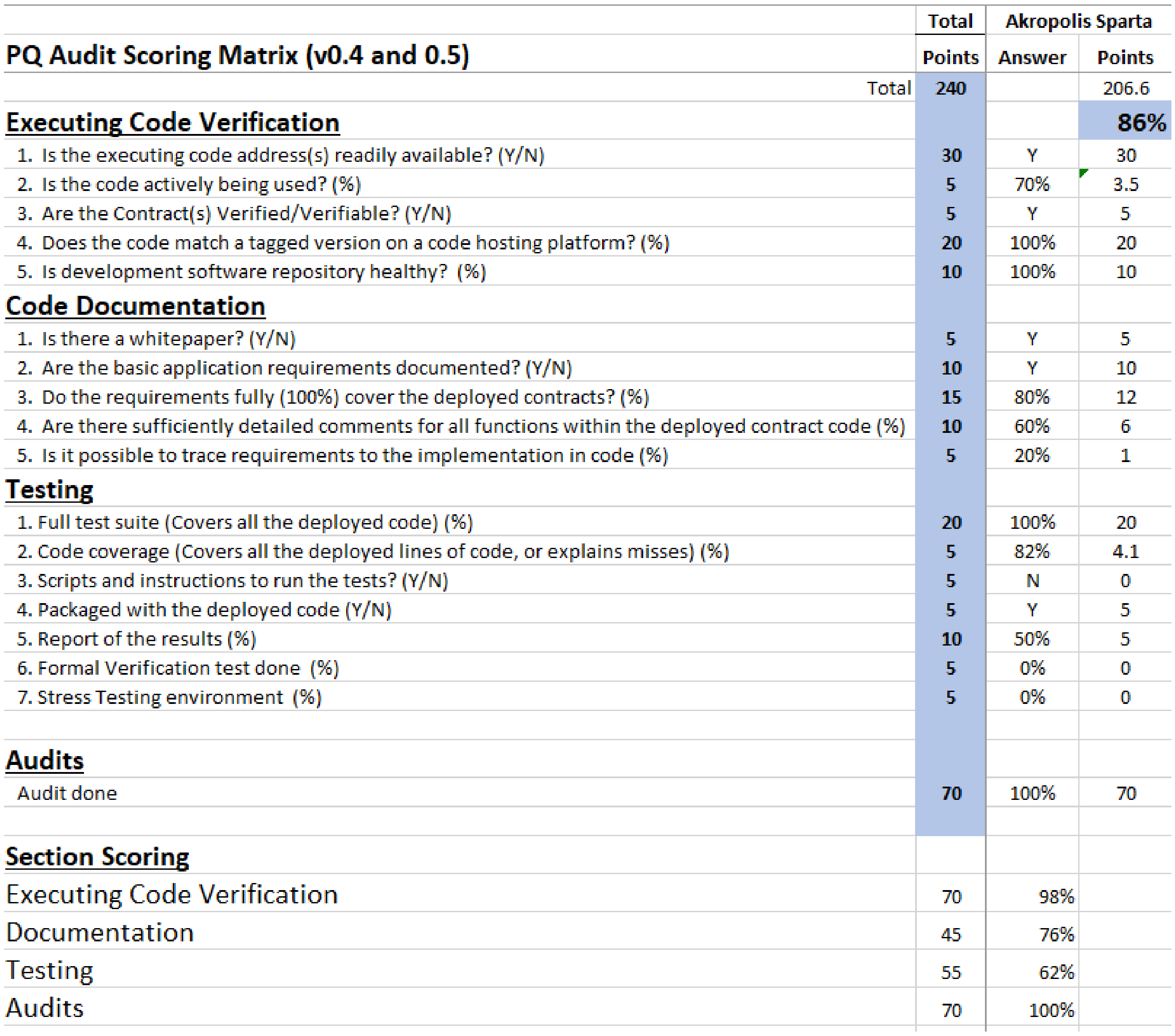
I started with Ethereum just before the DAO and that was a wonderful education. It showed the importance of code quality. The second Parity hack also showed the importance of good process. Here my aviation background offers some value. Aerospace knows how to make reliable code using quality processes.

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

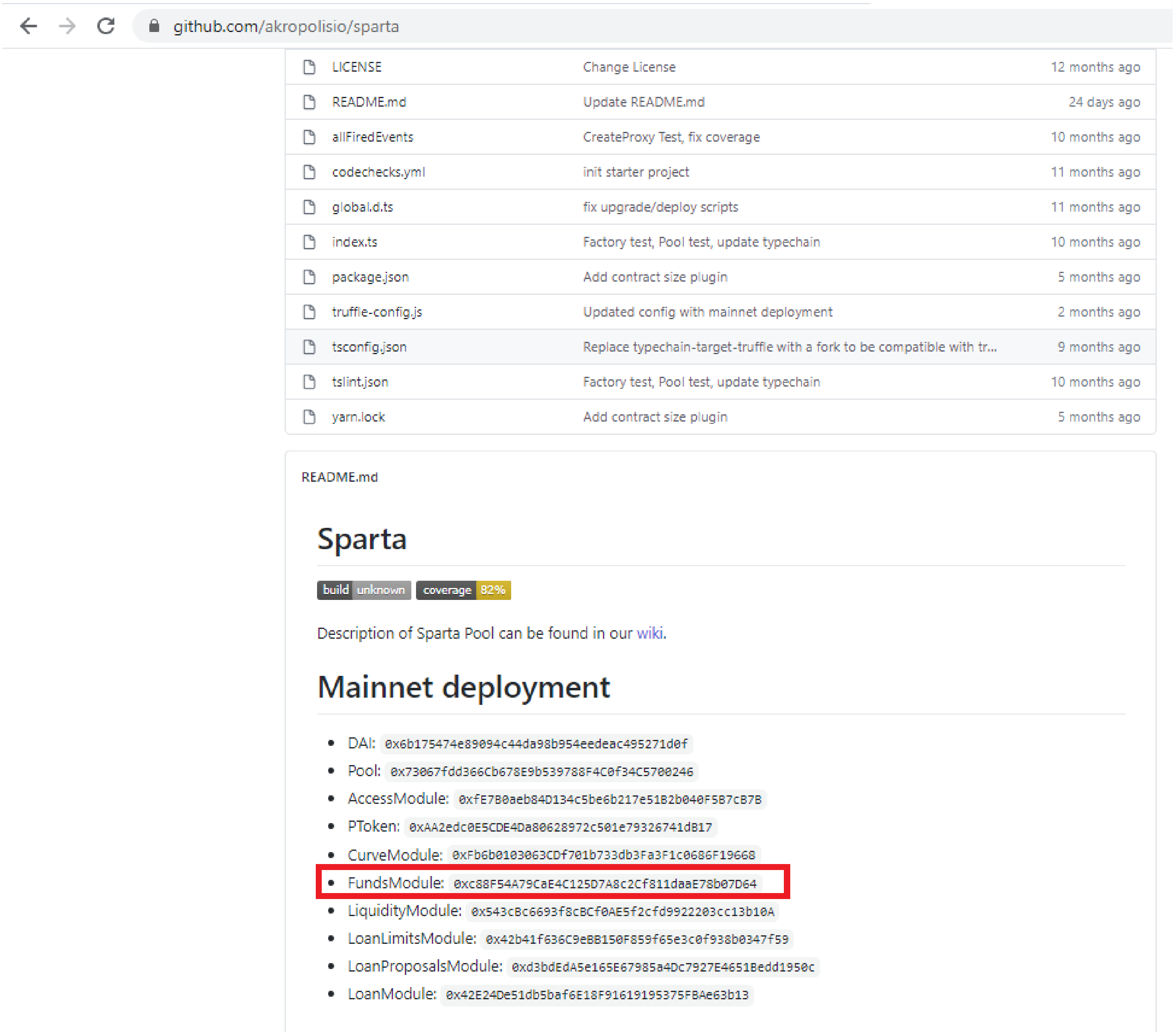
Career wise I am a business development manager for an avionics supplier.

Scoring Appendix



PQ Audit Scoring Matrix (v0.4 and 0.5)	Total Points	Akropolis Sparta Answer	Points
Total	240		206.6
Executing Code Verification			86%
1. Is the executing code address(s) readily available? (Y/N)	30	Y	30
2. Is the code actively being used? (%)	5	70%	3.5
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	100%	20
5. Is development software repository healthy? (%)	10	100%	10
Code Documentation			
1. Is there a whitepaper? (Y/N)	5	Y	5
2. Are the basic application requirements documented? (Y/N)	10	Y	10
3. Do the requirements fully (100%) cover the deployed contracts? (%)	15	80%	12
4. Are there sufficiently detailed comments for all functions within the deployed contract code (%)	10	60%	6
5. Is it possible to trace requirements to the implementation in code (%)	5	20%	1
Testing			
1. Full test suite (Covers all the deployed code) (%)	20	100%	20
2. Code coverage (Covers all the deployed lines of code, or explains misses) (%)	5	82%	4.1
3. Scripts and instructions to run the tests? (Y/N)	5	N	0
4. Packaged with the deployed code (Y/N)	5	Y	5
5. Report of the results (%)	10	50%	5
6. Formal Verification test done (%)	5	0%	0
7. Stress Testing environment (%)	5	0%	0
Audits			
Audit done	70	100%	70
Section Scoring			
Executing Code Verification	70	98%	
Documentation	45	76%	
Testing	55	62%	
Audits	70	100%	

Executing Code Appendix



github.com/akropolisio/sparta

LICENSE	Change License	12 months ago
README.md	Update README.md	24 days ago
allFiredEvents	CreateProxy Test, fix coverage	10 months ago
codechecks.yml	init starter project	11 months ago
global.d.ts	fix upgrade/deploy scripts	11 months ago
index.ts	Factory test, Pool test, update typechain	10 months ago
package.json	Add contract size plugin	5 months ago
truffle-config.js	Updated config with mainnet deployment	2 months ago
tsconfig.json	Replace typechain-target-truffle with a fork to be compatible with tr...	9 months ago
tslint.json	Factory test, Pool test, update typechain	10 months ago
yarn.lock	Add contract size plugin	5 months ago

README.md

Sparta

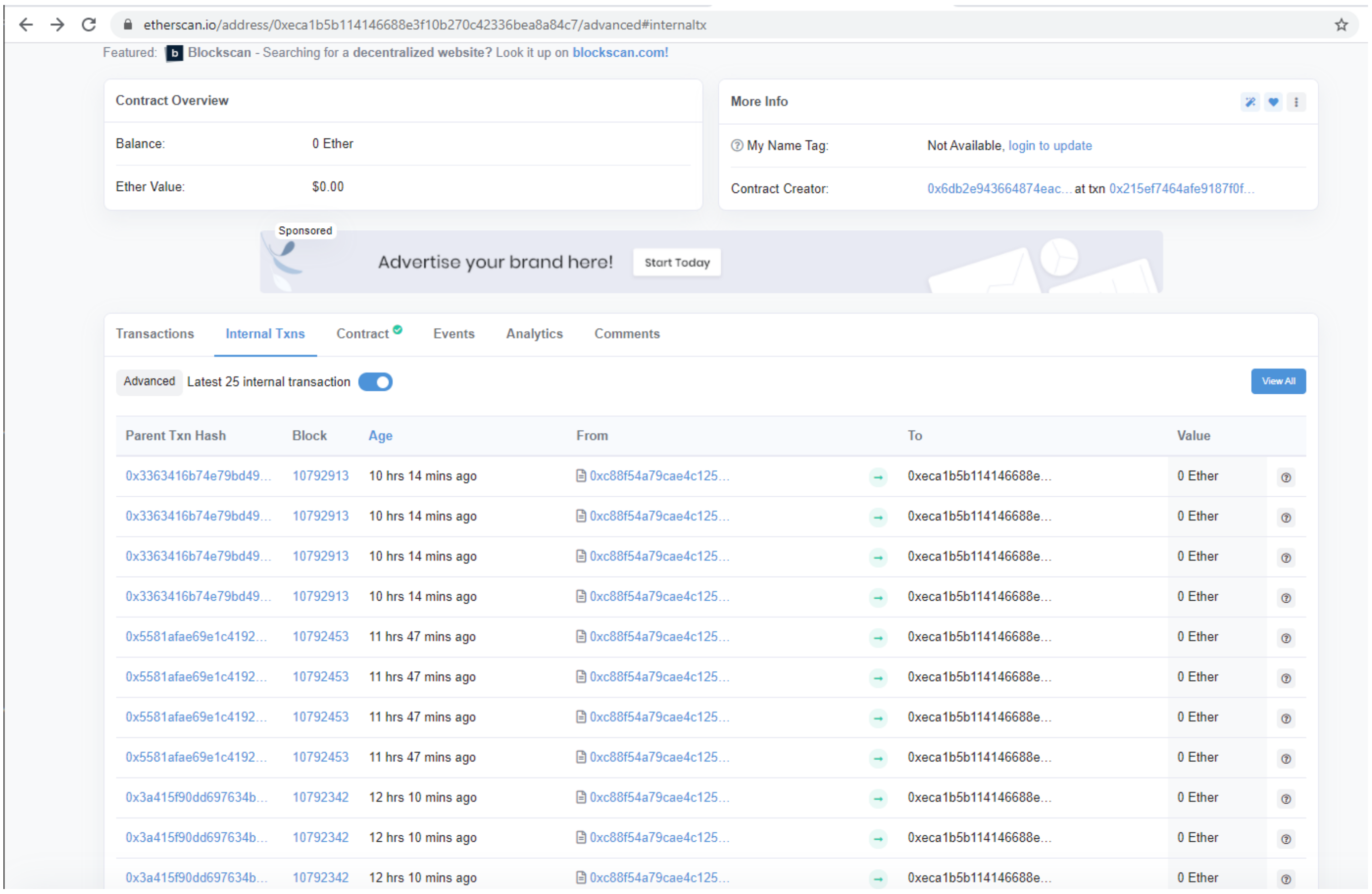
build unknown coverage 82%

Description of Sparta Pool can be found in our wiki.

Mainnet deployment

- DAI: `0x6b175474e89094c44da98b954eedeac495271d0f`
- Pool: `0x73067fdd366Cb678E9b539788F4C0f34C5700246`
- AccessModule: `0xfE7B0aeb84D134c5be6b217e51B2b040F5B7cB7B`
- PToken: `0xAA2edc0E5CDE4Da80628972c501e79326741dB17`
- CurveModule: `0xFb6b0103063CDf701b733db3Fa3F1c0686F19668`
- FundsModule: `0xc88F54A79CaE4C125D7A8c2Cf811daaE78b07D64`
- LiquidityModule: `0x543cBc6693f8cBCf0AE5f2cfd9922203cc13b10A`
- LoanLimitsModule: `0x42b41f636C9eBB150F859f65e3c0f938b0347f59`
- LoanProposalsModule: `0xd3bdEdA5e165E67985a4Dc7927E4651Bedd1950c`
- LoanModule: `0x42E24De51db5baf6E18F91619195375FBAe63b13`

Code Used Appendix



etherscan.io/address/0xeca1b5b114146688e3f10b270c42336bea8a84c7/advanced#internaltx

Featured: Blockscan - Searching for a decentralized website? Look it up on blockscan.com!

Contract Overview	
Balance:	0 Ether
Ether Value:	$0.00

More Info	
My Name Tag:	Not Available, login to update
Contract Creator:	0x6db2e943664874eac... at txn 0x215ef7464afe9187f0f...

Advanced Latest 25 internal transaction

Parent Txn Hash	Block	Age	From	To	Value
0x3363416b74e79bd49...	10792913	10 hrs 14 mins ago	0xc88f54a79cae4c125...	0xeca1b5b114146688e...	0 Ether
0x3363416b74e79bd49...	10792913	10 hrs 14 mins ago	0xc88f54a79cae4c125...	0xeca1b5b114146688e...	0 Ether
0x3363416b74e79bd49...	10792913	10 hrs 14 mins ago	0xc88f54a79cae4c125...	0xeca1b5b114146688e...	0 Ether
0x3363416b74e79bd49...	10792913	10 hrs 14 mins ago	0xc88f54a79cae4c125...	0xeca1b5b114146688e...	0 Ether
0x5581afae69e1c4192...	10792453	11 hrs 47 mins ago	0xc88f54a79cae4c125...	0xeca1b5b114146688e...	0 Ether
0x5581afae69e1c4192...	10792453	11 hrs 47 mins ago	0xc88f54a79cae4c125...	0xeca1b5b114146688e...	0 Ether
0x5581afae69e1c4192...	10792453	11 hrs 47 mins ago	0xc88f54a79cae4c125...	0xeca1b5b114146688e...	0 Ether
0x5581afae69e1c4192...	10792453	11 hrs 47 mins ago	0xc88f54a79cae4c125...	0xeca1b5b114146688e...	0 Ether
0x3a415f90dd697634b...	10792342	12 hrs 10 mins ago	0xc88f54a79cae4c125...	0xeca1b5b114146688e...	0 Ether
0x3a415f90dd697634b...	10792342	12 hrs 10 mins ago	0xc88f54a79cae4c125...	0xeca1b5b114146688e...	0 Ether
0x3a415f90dd697634b...	10792342	12 hrs 10 mins ago	0xc88f54a79cae4c125...	0xeca1b5b114146688e...	0 Ether

Example Code Appendix

```
//solhint-disable func-order
contract FundsModule is Module, IFundsModule, FundsOperatorRole {
    using SafeMath for uint256;
    uint256 private constant STATUS_PRICE_AMOUNT = 10**18;  // Used to calcu

    uint256 public lBalance;    //Tracked balance of liquid token, may be le
    mapping(address=>uint256) pBalances;    //Stores how many pTokens is loc

    function initialize(address _pool) public initializer {
        Module.initialize(_pool);
        FundsOperatorRole.initialize(_msgSender());
        //lBalance = lToken.balanceOf(address(this)); //We do not initialize
    }

    /**
     * @notice Deposit liquid tokens to the pool
     * @param from Address of the user, who sends tokens. Should have enough
     * @param amount Amount of tokens to deposit
     */
    function depositLTokens(address from, uint256 amount) public onlyFundsOp
        lBalance = lBalance.add(amount);
        require(lToken().transferFrom(from, address(this), amount), "FundsMo
        emitStatus();
    }

    /**
     * @notice Withdraw liquid tokens from the pool
     * @param to Address of the user, who sends tokens. Should have enough a
     * @param amount Amount of tokens to deposit
     */
    function withdrawLTokens(address to, uint256 amount) public onlyFundsOpe
        withdrawLTokens(to, amount, 0);
    }

    /**
     * @notice Withdraw liquid tokens from the pool
     * @param to Address of the user, who sends tokens. Should have enough a
     * @param amount Amount of tokens to deposit
     * @param poolFee Pool fee will be sent to pool owner
     */
    function withdrawLTokens(address to, uint256 amount, uint256 poolFee) pu
        lBalance = lBalance.sub(amount);
        if (amount > 0) { //This will be false for "fee only" withdrawal in
            require(lToken().transfer(to, amount), "FundsModule: withdraw fa
        }
        if (poolFee > 0) {
            lBalance = lBalance.sub(poolFee);
            require(lToken().transfer(owner(), poolFee), "FundsModule: fee t
        }
        emitStatus();
    }

    /**
     * @notice Deposit pool tokens to the pool
     * @param from Address of the user, who sends tokens. Should have enough
     * @param amount Amount of tokens to deposit
     */
    function depositPTokens(address from, uint256 amount) public onlyFundsOp
        require(pToken().transferFrom(from, address(this), amount), "FundsMo
        pBalances[from] = pBalances[from].add(amount);
    }

```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complexity
Solidity	22	2632	355	548	1459	131

Comments to Code 548 / 1459 = 38%

Typescript Tests

Language	Files	Lines	Blanks	Comments	Code	Complexity
TypeScript	16	2903	376	392	2135	100

Tests to Code 2135 / 1459 = 146%