0.7

Hegic V8888 Process Quality Review

Page: 31%

Overview

This is a Hegic Options V8888 Process Quality Review completed on 07/10/2021. 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 **31%**, 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

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.

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

Guidance:

Ethereum
Binance Smart Chain
Polygon
Avalanche
Terra
Celo
Arbitrum

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://hegic.gitbook.io/start/developers/contracts, as indicated in the Appendix.

Guidance:

100% Clearly labelled and on website, docs or repo, quick to find
70% Clearly labelled and on website, docs or repo but takes a bit of looking
40% Addresses in mainnet.json, in discord or sub graph, etc
20% Address found but labeling not clear or easy to find
0% Executing addresses could not be found

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

> **Answer:** 40%

Activity is 10 transactions a month on contract *Hegic WBTC Options Contract (mainnet)* 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/hegic/Hegic-protocol-v8888

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:** 0%

At just two branches and 23 commits, Hegic's development history is not one we'd hedge our bets with.

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:** No

Location: The anonymous Molly Wintermute is the creator of Hegic.

For a **"Yes"** in this question, the real names of some team members must be public on the website or other documentation (LinkedIn, etc). If the team is anonymous, 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://ipfs.io/ipfs/QmWy8x6vEunH4gD2gWT4Bt4bBwWX2KAEUov46tCLvMRcME

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

> ⚠ **Answer:** No

There are no software functions in the Hegic 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%

Contracts are identified, but there is no software function documentation. The diagram provides insufficient detail to explain how the contracts operate.

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:** 57%

Code examples are in the Appendix. As per the SLOC, there is 57% 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%

The documentation lists the functions but does not describe them.

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:** 100%

Code examples are in the Appendix. As per the SLOC, there is 194% 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	

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

> **Answer:** 47%

Tests for code coverage gave back 47.36%.

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 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/Instructions location: https://github.com/hegic/Hegic-protocol-v8888

14) Report of the results (%)

> **Answer:** 0%

No test report was 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:** 100%

Hegic is deployed to Ropsten testnet.

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%

No audits have taken place on Hegic V8888. Older versions (V888) have been audited by PeckShield, but since V8888 is a different product this iteration remains unaudited.

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:** 0%

Hegic announced an intention to launch a bug bounty program, but no follow-up was found.

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:** 0%

There is no access control information. For older versions, admin information was found but this must be updated in V888 documentation in order to remain relevant.

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%

No V8888 relevant information was found.

Guidance:

All the contracts are immutable -- 100% OR

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

How to improve this score:

Create a document that covers the items described above. An example is enclosed.

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

> ⚠ **Answer:** 0%

No information was found.

Guidance:

100%	All the contracts are immutable
90%	Description relates to investments safety and updates in clear, complete non-software l language
30%	Description all in software specific language
0%	No admin control information could not be found

How to improve this score:

Create a document that covers the items described above in plain language that investors can understand. An example is enclosed.

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

> ⚠ **Answer:** 0%

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

PQ Audit Scoring Matrix (v0.7)	Total Points	Hegic V8888 Answer	Hegic V8888 Points
Total	260		81.2
Code and Team			31%
1) Are the executing code addresses readily available? (%)	20	100%	20
2) Is the code actively being used? (%)	5	40%	2
3) Is there a public software repository? (Y/N)	5	y	5
4) Is there a development history visible? (%)	5	0%	0
5) Is the team public (not anonymous)? (Y/N)	15	n	0
Code Documentation			
6) Is there a whitepaper? (Y/N)	5	y	5
7) Are the basic software functions documented? (Y/N)	10	n	0
8) Does the software function documentation fully (100%) cov	15	0%	0
9) Are there sufficiently detailed comments for all functions w	5	57%	2.85
10) Is it possible to trace from software documentation to the	10	0%	0
Testing			
11) Full test suite (Covers all the deployed code) (%)	20	100%	20
12) Code coverage (Covers all the deployed lines of code, or ex	5	47%	2.35
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	100%	5
Security			
17) Did 3rd Party audits take place? (%)	70	20%	14
18) Is the bug bounty acceptable high? (%)	10	0%	0
Access Controls			
19) Can a user clearly and quickly find the status of the admin	5	0%	0
20) Is the information clear and complete	10	0%	0
21) Is the information in non-technical terms	10	0%	0
22) Is there Pause Control documentation including records of	10	0%	0
Section Scoring			
Code and Team	50	54%	
Documentation	45	17%	
Testing	50	65%	
Security	80	18%	
Access Controls	35	0%	

Executing Code Appendix

Hegic WBTC Options Contract (mainnet):

```
0x3961245db602ed7c03eeccda33ea3846bd8723bd
```

Hegic ETH Options Contract (mainnet):

```
0xEfC0eEAdC1132A12c9487d800112693bf49EcfA2
```

Hegic WBTC Bidirectional Liquidity Pool Contract (mainnet):

```
0x20DD9e22d22dd0a6ef74a520cb08303B5faD5dE7
```

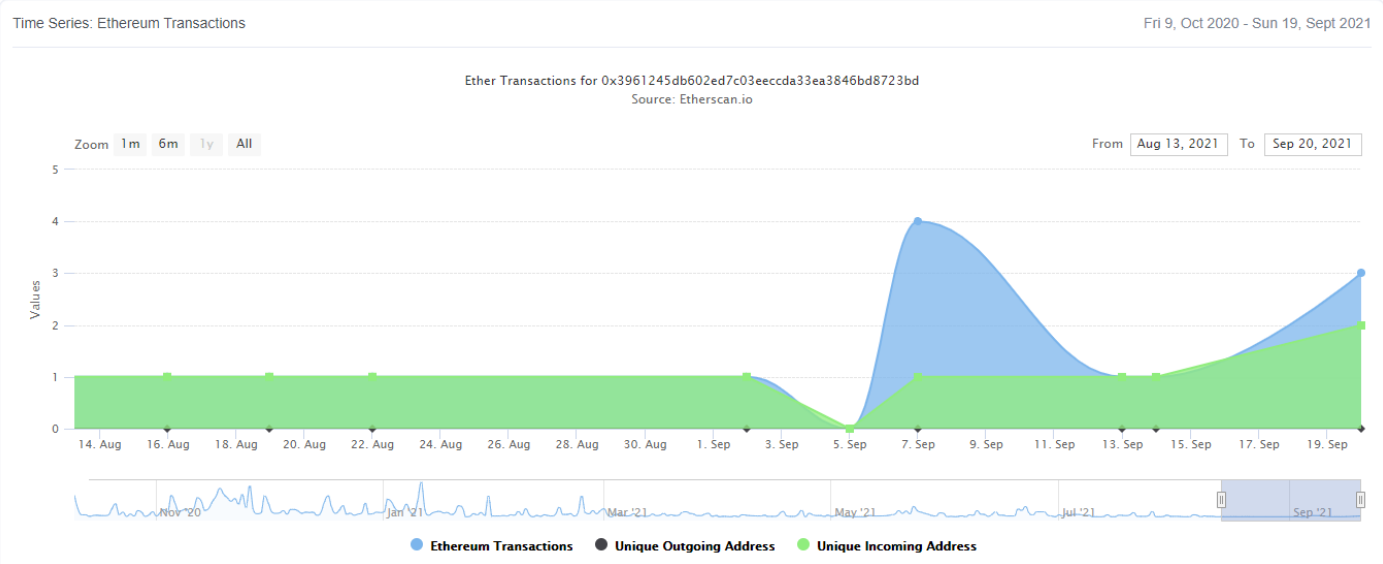
Hegic ETH Bidirectional Liquidity Pool Contract (mainnet):

```
0x878F15ffC8b894A1BA7647c7176E4C01f74e140b
```

Hegic WBTC Staking Lots Contract (mainnet):

```
0x840a1AE46B7364855206Eb5b7286Ab7E207e515b
```

Code Used Appendix



Example Code Appendix

```
1 import "../Interfaces/Interfaces.sol";
2 import "../Interfaces/IOptionsManager.sol";
3
4 /**
5  * @author 0mllwntrmt3
6
```

```
   * @title Hegic Protocol V8888 Facade Contract
 7 * @notice The contract that calculates the options prices,
 8 * conducts the process of buying options, converts the premiums
 9 * into the token that the pool is denominated in and grants
10 * permissions to the contracts such as GSN (Gas Station Network).
11 **/
12
13 contract Facade is Ownable {
14     using SafeERC20 for IERC20;
15
16     IWETH public immutable WETH;
17     IUniswapV2Router01 public immutable exchange;
18     IOptionsManager public immutable optionsManager;
19     address public _trustedForwarder;
20
21     constructor(
22         IWETH weth,
23         IUniswapV2Router01 router,
24         IOptionsManager manager,
25         address trustedForwarder
26     ) {
27         WETH = weth;
28         exchange = router;
29         _trustedForwarder = trustedForwarder;
30         optionsManager = manager;
31     }
32
33     /**
34      * @notice Used for calculating the option price (the premium) and using
35      * the swap router (if needed) to convert the tokens with which the user
36      * pays the premium into the token in which the pool is denominated.
37      * @param period The option period
38      * @param amount The option size
39      * @param strike The option strike
40      * @param total The total premium
41      * @param baseTotal The part of the premium that
42      * is distributed among the liquidity providers
43      * @param settlementFee The part of the premium that
44      * is distributed among the HEGIC staking participants
45      **/
46     function getOptionPrice(
47         IHegicPool pool,
48         uint256 period,
49         uint256 amount,
50         uint256 strike,
51         address[] calldata swappath
52     )
53         public
54         view
55         returns (
56             uint256 total,
57             uint256 baseTotal,
58             uint256 settlementFee,
```

```
            uint256 premium
        )
    {
        (uint256 _baseTotal, uint256 baseSettlementFee, uint256 basePremium) =
            getBaseOptionCost(pool, period, amount, strike);
        if (swappath.length > 1)
            total = exchange.getAmountsIn(_baseTotal, swappath)[0];
        else total = _baseTotal;

        baseTotal = _baseTotal;
        settlementFee = (total * baseSettlementFee) / baseTotal;
        premium = (total * basePremium) / baseTotal;
    }

    /**
     * @notice Used for calculating the option price (the premium)
     * in the token in which the pool is denominated.
     * @param period The option period
     * @param amount The option size
     * @param strike The option strike
     **/
    function getBaseOptionCost(
        IHegicPool pool,
        uint256 period,
        uint256 amount,
        uint256 strike
    )
        public
        view
        returns (
            uint256 total,
            uint256 settlementFee,
            uint256 premium
        )
    {
        (settlementFee, premium) = pool.calculateTotalPremium(
            period,
            amount,
            strike
        );
        total = premium + settlementFee;
    }

    /**
     * @notice Used for approving the pools contracts addresses.
     **/
    function poolApprove(IHegicPool pool) external {
        pool.token().safeApprove(address(pool), 0);
        pool.token().safeApprove(address(pool), type(uint256).max);
    }

    /**
     * @notice Used for buying the option contract and converting
```

```
     * the buyer's tokens (the total premium) into the token
     * in which the pool is denominated.
     * @param period The option period
     * @param amount The option size
     * @param strike The option strike
     * @param acceptablePrice The highest acceptable price
     **/
    function createOption(
        IHegicPool pool,
        uint256 period,
        uint256 amount,
        uint256 strike,
        address[] calldata swappath,
        uint256 acceptablePrice
    ) external {
        address buyer = _msgSender();
        (uint256 optionPrice, uint256 rawOptionPrice, , ) =
            getOptionPrice(pool, period, amount, strike, swappath);
        require(
            optionPrice <= acceptablePrice,
            "Facade Error: The option price is too high"
        );
        IERC20 paymentToken = IERC20(swappath[0]);
        paymentToken.safeTransferFrom(buyer, address(this), optionPrice);
        if (swappath.length > 1) {
            if (
                paymentToken.allowance(address(this), address(exchange)) <
                optionPrice
            ) {
                paymentToken.safeApprove(address(exchange), 0);
                paymentToken.safeApprove(address(exchange), type(uint256).max);
            }

            exchange.swapTokensForExactTokens(
                rawOptionPrice,
                optionPrice,
                swappath,
                address(this),
                block.timestamp
            );
        }
        pool.sellOption(buyer, period, amount, strike);
    }

    /**
     * @notice Used for converting the liquidity provider's Ether (ETH)
     * into Wrapped Ether (WETH) and providing the funds into the pool.
     * @param hedged The liquidity tranche type: hedged or unhedged (classic)
     **/
    function provideEthToPool(
        IHegicPool pool,
        bool hedged,
        uint256 minShare
```

```
165     ) external payable returns (uint256) {
166         WETH.deposit{value: msg.value}();
167         if (WETH.allowance(address(this), address(pool)) < msg.value)
168             WETH.approve(address(pool), type(uint256).max);
169         return pool.provideFrom(msg.sender, msg.value, hedged, minShare);
170     }
171 
172     /**
173      * @notice Unlocks the array of options.
174      * @param optionIDs The array of options
175      **/
176     function unlockAll(IHegicPool pool, uint256[] calldata optionIDs) external {
177         uint256 arrayLength = optionIDs.length;
178         for (uint256 i = 0; i < arrayLength; i++) {
179             pool.unlock(optionIDs[i]);
180         }
181     }
182 
183     /**
184      * @notice Used for granting the GSN (Gas Station Network) contract
185      * the permission to pay the gas (transaction) fees for the users.
186      * @param forwarder GSN (Gas Station Network) contract address
187      **/
188     function isTrustedForwarder(address forwarder) public view returns (bool) {
189         return forwarder == _trustedForwarder;
190     }
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complex
Solidity	8	1331	111	441	779	46

Comments to Code 441/779 = 57%

TypeScript Tests

Language	Files	Lines	Blanks	Comments	Code	Complex
TypeScript	9	2666	157	1001	1508	8

Tests to Code 1508/779 = 194%