Gnosis Safe PQ Review

Score: 95%

This is a Gnosis Safe Process Quality Review completed on 23 February, 2021. It was performed using the Process Review process (version 0.6.2) and is documented here. The review was performed by ShinkaRex of DeFiSafety. Check out our Telegram.

The final score of the review is 95%, a stellar 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**

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.

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? (Y/N)
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)

Are the executing code addresses readily available? (Y/N)

> Answer: Yes

Gnosis safe is an app rather than a protocol. When each safe is created the contract address is given. We will give a Yes for this scoring.

Is the code actively being used? (%)

> Answer: 100%

Once again, as Gnosis Safe is an app this question does not really apply. For this reason a 100% score is given. Each safe is used as much as its owners want.

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

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

> Answer: Yes

GitHub: https://github.com/gnosis/safe-contracts

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.

Is there a development history visible? (%)

> Answer: 100%

With 500+ commits and 12+ branches, this is a very 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

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.

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

> Answer: Yes

While there is not a teams section, the actual names are used in the GitHub and documentation. the Gnosis team are quite public.

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;

1. Is there a whitepaper? (Y/N)
2. Are the basic software functions documented? (Y/N)
3. Does the software function documentation 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 from software documentation to the implementation in codee (%)

Is there a whitepaper? (Y/N)

Answer: Yes

Location: **https://docs.gnosis.io/safe/docs/intro_assets/**

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

Answer: Yes

Location: https://docs.gnosis.io/safe/docs/contracts_details/

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

Answer: 100%

Documentation appears complete, covering all functions.

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

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

Answer: 67

Code examples are in the Appendix. As per the SLOC, there is 47% 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.

Is it possible to trace from software documentation to the implementation in code (%)

> ⚠ Answer: 0%

No connection between documentation and 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 to Code ratio is 339%

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 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: 98%

The latest full test coverage showed 98% (https://coveralls.io/builds/37631390).

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: Yes

Packaged with the deployed code (Y/N)

Answer: Yes

Report of the results (%)

Answer: 70%

No test report besides coverall report (https://coveralls.io/builds/37631390) visible.

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.

Formal Verification test done (%)

> Answer: 100%

Formal verification reports at https://github.com/gnosis/safe-contracts/blob/v1.2.0/docs/Gnosis_Safe_Formal_Verification_Report_1_0_0.pdf.

Stress Testing environment (%)

> Answer: 100%

Rinkeby test interfaces are available; https://docs.gnosis.io/safe/docs/intro_interfaces/

Audits

> Answer: 100%

V1.2 audit: https://github.com/gnosis/safe-contracts/blob/v1.2.0/docs/Gnosis_Safe_Audit_Report_1_2_0.pdf

V1.1 audit https://github.com/gnosis/safe-contracts/blob/v1.1.1/docs/audit_1_1_1.md

Guidance:

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

Appendices

Author Details

The author of this review is Rex of 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 (v0.6)	Total Points	Gnosis Safe Answer	Gnosis Safe Points
Total	240		228.6
Code and Team			95%
1. Are the executing code addresses readily available? (Y/N)	30	Y	30
2. Is the code actively being used? (%)	10	100%	10
3. Is there a public software repository? (Y/N)	5	Y	5
4. Is there a development history visible? (%)	5	100%	5
Is the team public (not anonymous)? (Y/N)	20	Y	20
Code Documentation			
1. Is there a whitepaper? (Y/N)	5	Y	5
2. Are the basic software functions documented? (Y/N)	10	Y	10
3. Does the software function documentation fully (100%) cover the deployed contracts? (%)	15	100%	15
4. Are there sufficiently detailed comments for all functions within the deployed contract code (%)	10	67%	6.7
5 Is it possible to trace from software documentation to the implementation in code (%)	5	0%	0
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	98%	4.9
3. Scripts and instructions to run the tests? (Y/N)	5	Y	5
4. Packaged with the deployed code (Y/N)	5	Y	5
5. Report of the results (%)	10	70%	7
6. Formal Verification test done (%)	5	100%	5
7. Stress Testing environment (%)	5	100%	5
Audits			
Audit done	70	100%	70
Section Scoring			
Code and Team	70	100%	
Documentation	45	82%	
Testing	55	94%	
Audits	70	100%	

Example Code Appendix

```
1  // @title Gnosis Safe - A multisignature wallet with support for confirmati
2  /// @author Stefan George - <stefan@gnosis.io>
3  /// @author Richard Meissner - <richard@gnosis.io>
4  /// @author Ricardo Guilherme Schmidt - (Status Research & Development GmbH)
5  contract GnosisSafe
6      is MasterCopy, ModuleManager, OwnerManager, SignatureDecoder, SecuredTol
7
8      using GnosisSafeMath for uint256;
9
10     string public constant NAME = "Gnosis Safe";
11     string public constant VERSION = "1.2.0";
12
13     //keccak256(
14     //    "EIP712Domain(address verifyingContract)"
15     //);
16     bytes32 private constant DOMAIN_SEPARATOR_TYPEHASH = 0x035aff83d86937d35
17
```

```
18      //keccak256(
19      //    "SafeTx(address to,uint256 value,bytes data,uint8 operation,uint25
20      //);
21      bytes32 private constant SAFE_TX_TYPEHASH = 0xbb8310d486368db6bd6f84940
22
23      //keccak256(
24      //    "SafeMessage(bytes message)"
25      //);
26      bytes32 private constant SAFE_MSG_TYPEHASH = 0x60b3cbf8b4a223d68d641b3b
27
28      event ApproveHash(
29          bytes32 indexed approvedHash,
30          address indexed owner
31      );
32      event SignMsg(
33          bytes32 indexed msgHash
34      );
35      event ExecutionFailure(
36          bytes32 txHash, uint256 payment
37      );
38      event ExecutionSuccess(
39          bytes32 txHash, uint256 payment
40      );
41
42      uint256 public nonce;
43      bytes32 public domainSeparator;
44      // Mapping to keep track of all message hashes that have been approve by
45      mapping(bytes32 => uint256) public signedMessages;
46      // Mapping to keep track of all hashes (message or transaction) that hav
47      mapping(address => mapping(bytes32 => uint256)) public approvedHashes;
48
49      // This constructor ensures that this contract can only be used as a mas
50      constructor() public {
51          // By setting the threshold it is not possible to call setup anymore
52          // so we create a Safe with 0 owners and threshold 1.
53          // This is an unusable Safe, perfect for the mastercopy
54          threshold = 1;
55      }
56
57      /// @dev Setup function sets initial storage of contract.
58      /// @param _owners List of Safe owners.
59      /// @param _threshold Number of required confirmations for a Safe transa
60      /// @param to Contract address for optional delegate call.
61      /// @param data Data payload for optional delegate call.
62      /// @param fallbackHandler Handler for fallback calls to this contract
63      /// @param paymentToken Token that should be used for the payment (0 is
64      /// @param payment Value that should be paid
65      /// @param paymentReceiver Adddress that should receive the payment (or
66      function setup(
67          address[] calldata _owners,
68          uint256 _threshold,
69          address to,
70          bytes calldata data,
71          address fallbackHandler,
72          address paymentToken,
```

```
73          uint256 payment,
74          address payable paymentReceiver
75      )
76          external
77      {
78          require(domainSeparator == 0, "Domain Separator already set!");
79          domainSeparator = keccak256(abi.encode(DOMAIN_SEPARATOR_TYPEHASH, tl
80          setupOwners(_owners, _threshold);
81          if (fallbackHandler != address(0)) internalSetFallbackHandler(fallb
82          // As setupOwners can only be called if the contract has not been i
83          setupModules(to, data);
84
85          if (payment > 0) {
86              // To avoid running into issues with EIP-170 we reuse the handl
87              // baseGas = 0, gasPrice = 1 and gas = payment => amount = (pay
88              handlePayment(payment, 0, 1, paymentToken, paymentReceiver);
89          }
90      }
91
```

SLOC Appendix

Solidity Contracts

Language	Files	Lines	Blanks	Comments	Code	Complexity
Solidity	12	1014	91	294	629	105

Comments to Code 294 / 629 = 46%

Typecript Tests

Language	Files	Lines	Blanks	Comments	Code	Complexity
TypeScript	23	2599	416	52	2131	97

Tests to Code 2131 / 629 = 339%