DEFISAFETY

Pre-Audit Review

Question List v0.2

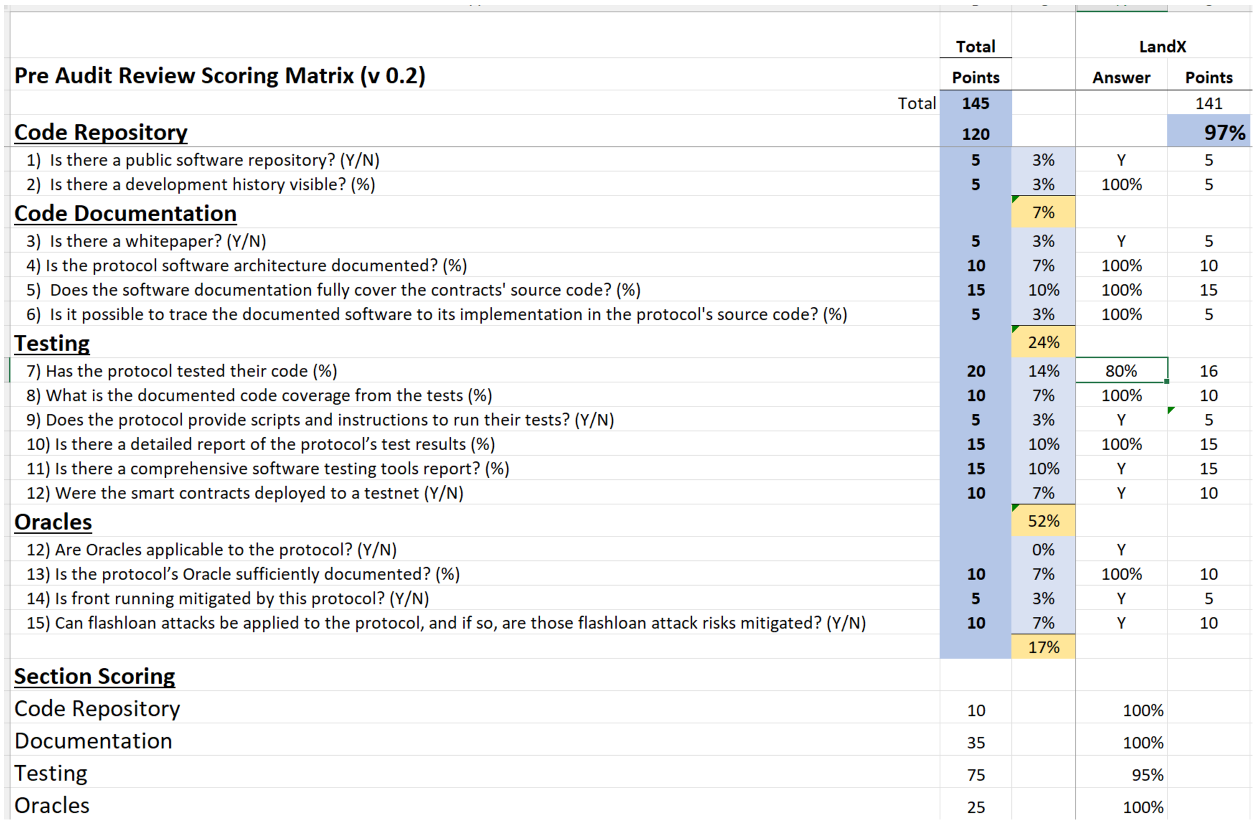
1. Introduction

This documents the DeFiSafety Pre-Audit Reviews for LandX

A Pre-Audit Review (PAR) is a quick review that determines if a protocol is ready for an Audit. It checks the software documentation, testing, tools and other aspects and creates a score.

This document contains the questions used by the process to create the score.

Total Score: 97%



	Total		LandX	
Pre Audit Review Scoring Matrix (v 0.2)	Points		Answer	Points
Total	145			141
Code Repository	120			97%
1) Is there a public software repository? (Y/N)	5	3%	Y	5
2) Is there a development history visible? (%)	5	3%	100%	5
Code Documentation		7%		
3) Is there a whitepaper? (Y/N)	5	3%	Y	5
4) Is the protocol software architecture documented? (%)	10	7%	100%	10
5) Does the software documentation fully cover the contracts' source code? (%)	15	10%	100%	15
6) Is it possible to trace the documented software to its implementation in the protocol's source code? (%)	5	3%	100%	5
Testing		24%		
7) Has the protocol tested their code (%)	20	14%	80%	16
8) What is the documented code coverage from the tests (%)	10	7%	100%	10
9) Does the protocol provide scripts and instructions to run their tests? (Y/N)	5	3%	Y	5
10) Is there a detailed report of the protocol's test results (%)	15	10%	100%	15
11) Is there a comprehensive software testing tools report? (%)	15	10%	Y	15
12) Were the smart contracts deployed to a testnet (Y/N)	10	7%	Y	10
Oracles		52%		
12) Are Oracles applicable to the protocol? (Y/N)		0%	Y	
13) Is the protocol's Oracle sufficiently documented? (%)	10	7%	100%	10
14) Is front running mitigated by this protocol? (Y/N)	5	3%	Y	5
15) Can flashloan attacks be applied to the protocol, and if so, are those flashloan attack risks mitigated? (Y/N)	10	7%	Y	10
		17%		
Section Scoring				
Code Repository	10		100%	
Documentation	35		100%	
Testing	75		95%	
Oracles	25		100%	

1.1. Disclaimer

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, 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. 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.

1.2. 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 that I'd like audited
- 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 admin admin controls and how they're used
- This is where the protocol gets its data, and these are some proactive steps we've taken to avoid problems.

1.3. Sections of the Review Process

The process breaks the scores into the following sections:

- GitHub –
- Code Documentation -- Software Documentation for the Deployed Code
- Test -- Overall testing measures for the Deployed Code
- Admin controls -- Review of the relevant information about admin controls
- Oracles -- Explanation of the data sources used by the protocol and identification of potential vulnerabilities

1.4. Scoring

A review's Final Score is indicated as a percentage. This percentage is calculated as total Achieved Points divided by the total Possible Points. For each question the answer can be either yes (Y), no (N), or a percentage (%). Each of these questions has a "Scoring Weight", as some are more important than others. For example, "Question 1" is more important than "Question 15", and therefore has a higher weight.

The individual question's Achieved Points is the Scoring Weight times the answer (yes, no, %). The review's Total Achieved Points is the sum of every question's points. For our purposes, a passing score is one that receives 70% or more.

2. GitHub

Any PAR starts with the code which the protocol uses, and how frequently it is interacted with by users.

The score for this section is derived from the individual scores of the following questions:

1) Is the scope of the audit defined to a GitHub repository? (Y/N)
2) Is there a development history visible? (%)

2.1. Q1 – Is the scope of the audit defined to a GitHub repository? (Y/N)

For this metric, we look at the identification of the scope of the protocol's software repository. A defined software repository is essential for auditors, as it enables them to go through it and read any relevant smart contract code. This practice enables a lot of specificity between the deployed contracts and the source code, and subsequently provides increased speed for there auditor.

Scoring: Y (Y/N)

If a protocol's specified GitHub is visible, even one just made for the audit, then this scores as "Yes". For teams with no specified audit repository, then "No".

2.2. Q2 - Is there a development history visible? (%)

This metric checks a protocol's software repository for a robust development history. We measure this by counting GitHub commits, branches and releases in a software repository. All of these act as a ledger of edits, collaboration, and general workflow.

Scoring: 100 (%)

Notes: Current repository displays 126 commits and 3 branches

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 11+ commits, 3+branches

0% Less than 2 branches or less than 10 commits

3. Code Documentation

The documentation section analyses the quality of the protocol's software documentation. For these metrics, we will be looking at the general accessibility of the documentation, as well as verifying if the functionalities of the software are explained or not. This iteration of the Process Audit standard requests only basic documentation, although increasingly detailed documentation will naturally score higher.

The score for this section is derived from the individual scores of the following questions:

3) Is there a whitepaper? (Y/N)
4) Is the protocol software architecture documented? (%)
5) Does the software documentation fully cover the deployed contracts' source code? (Y/N)
6) Is it possible to trace the documented software to its implementation in the protocol's source code? (%)

3.1. Q3 - Is there a whitepaper? (Y/N)

Is there a white paper or other basic (or complex) description of what the project is doing referenced to by either the protocol's website or GitHub? In addition, Medium articles, GitBooks, or GitHub README.md explaining the application are all admissible.

Scoring: Y (Y/N)

Notes: There is a GitHub document explaining smart contracts here: https://landxit.github.io/land-x-smart-contracts/

3.2. Q4 - Is the protocol software architecture documented? (%)

This score requires a section of the documentation that specifically covers the protocol's architecture. Architecture is a loose term that boils down to a section that details "software function used in contract (code) + what it does/how it does it". In addition, this can also be presented as a diagram that may include the following:

- Arrows indicating how the smart contracts interact with each other
- Specific reference to the software functions themselves
- A written explanation on how the smart contracts interact alongside the directional arrows.
- Formal Requirements

Scoring: 100 (%)

Notes: Software architecture is documented through a diagram here: https://landxit.github.io/land-x-smart-contracts/#landx-smart-contract-interaction-overview . Written explanations can also be found on the same page.

Guidance:

100% Clear complete architecture docs and formal requirements

70% Clear complete architecture docs

50% Clear complete architecture on major parts of protocol

30% Simple architecture docs that covers part of the protocol

0% No software architecture documentation

3.3. Q5 - Does the software documentation fully cover the relevant contracts' source code? (%)

This score requires documentation specifically written about a protocol's (audit scope) smart contract source code. As such, any generalized math formulas or state diagrams without directly referencing the code do not count towards this score.

3.3.1. Scoring

Scoring: 100 (%)

Notes: Software architecture is documented through a diagram here: https://landxit.github.io/land-x-smart-contracts/#landx-smart-contract-interaction-overview

Guidance:

100%	All contracts and functions documented
80%	Only the major functions documented
79-1%	Estimate of the level of software documentation
20%	Only API function documented
0%	No software documentation

3.3.2. Q6 - Is it possible to trace the documented software to its implementation in the protocol's source code? (%)

In order to meet our standards, every piece of deployed code must have 100% traceability. For reference, check the SecurEth guidelines on traceability.

Scoring: 100 (%)

Notes: All contract functions are documented here: https://landxit.github.io/land-x-smart-contracts/#contracts-functions-description . The descriptions are complete and source code is displayed. Additionally, explicit traceability is implemented as hyperlinks redirect to the source code in the GitHub.

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

0% - No connection between documentation and code

4. Testing

This section covers the testing process of the protocol's smart contract code before its deployment on the mainnet. The testing suite should be in the same GitHub repository as the contracts themselves. This suite also should completely cover all the code in the deployed contracts and come with a detailed report that indicates a successful test run. This is often referred to as a "code coverage" test. Any additional testing can include a Formal Verification test as well as active test/stress environments (on a chain-specific testnet like Kovan for ethereum, for example).

The score for this section is derived from the individual scores of the following questions:

- Has the protocol tested their code? (%)
- How covered is the protocol's code? (%)
- Does the protocol provide scripts and instructions to run tests? (Y/N)
- Is there a report of the protocol's test results? (%)
- Which tools does the protocol use to test their smart contracts?
- Were the smart contracts deployed to a testnet? (Y/N)

4.1. Q7 - Has the protocol tested their code? (%)

Scoring: 80 (%)

Notes: TtC is recorded at 75% (1743 test lines over 2306 lines) if we count all the files in the contract directory, given a score of 40% (which gives a 92% final score. Or we ignore the files that are ignored for code coverage, implying they are not tested. You get a TtC of 113%, giving an 80% score and over all score of 97%. The real question of course is, Why are you not testing these files and should you?

Files not code covered: LNDX.sol, LNDXGovernor, OracleMulti, usdc, veLNDX.sol

4.1.1. Scoring

For our purposes, this score is guided by the Test to Code ratio (TtC), which is calculated by dividing the lines of code in the deployed smart contracts by the lines of code in the protocol's testing suite. A good TtC ratio is over 100%, which means that approximately every line of deployed code has undergone some form of testing, which makes for better code. An ideal TtC would be 120%, as this proves a rigorous testing process. However, the reviewer's 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 found

4.1.2. 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. Ideally, you should have a test file per smart contract deployed on the blockchain. In addition, providing scripts to run your tests is an essential part of testing transparency, and no testing suite should come without them.

4.2. Q8 - What is the documented code coverage from the tests? (%)

4.2.1. Key Specifications:

- A code coverage test can be performed by the protocol itself, a third-party auditor, or a code coverage service such as Coveralls.
- Ideally, a coverage test should include the coverage of both lines of code and branches on the GitHub.

Scoring: 100 (%)

Notes: LandX has a code coverage badge found in the repository's README.md displaying 100%.

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 as identified by the TtC being greater than 120%
30% - Some tests evident (TtC between 60% to 120%) but not complete 0% - No test for coverage seen
0% - No evidence of code coverage and inadequate number of tests (TtC < 60%)

4.2.2. How to improve this score

This score can be improved by performing code coverage tests that are as close to 100% coverage as possible. If some lines of code or entire contracts are missed, you should clearly outline why this is the case in your coverage report. Hence, you should also be aiming to perform code coverage tests upon every single deployment. This proves that the code is rigorously tested, and therefore has a degree of reliability attributed to it. Integrate the result of your code coverage result in your GitHub repository readme, like Tetu protocol has done here.

4.3. Q9 - Does the protocol provide scripts and instructions to run their tests? (Y/N)

4.3.1. Key Specifications:

- Although we give points for the presence of testing scripts OR instructions on how to compile them in the proper environment, it is best practice to have both ready and available.
- Scripts can be seen as an implicit way to permit users to run the same tests that a specific protocol has. However, not everyone knows how to run scripts without instructions, and that is why instructions to run the tests and/or the scripts are an important part of it as well.

4.3.2. Scoring

Scoring: Y (Y/N)

Notes Testing instructions can be found here: https://landxit.github.io/land-x-smart-contracts/#build-and-test

Guidance:
Yes - Scripts and/or instructions to run tests are available in the testing suite
No - Scripts and/or instructions to run tests are not available in the testing suite

4.4. Q10 - Is there a detailed report of the protocol's test results?(%)

Scoring: Y (Y/N)

Notes A detailed test report could be found in the Actions' tab of the CI here: https://github.com/LandXit/land-x-smart-contracts/actions/runs/3396718433/jobs/5648090183

Guidance:
100% - Detailed test report as described below
50% - Code coverage report visible
0% - No test report evident

4.4.1. How to improve this score

Add a code coverage test report with the results. This should not only be a code coverage output, but rather a combination of your coverage output and a deeper insight on your methodology used. An exemplary test report from Balancer Finance can be found here.

4.5. Q11 – Is a tool report available?

Scoring: Y (Y/N)

Notes LandX uses Ganache as a testing tool set.

Guidance:
Yes – Tools report was documented, and the full report is readily available
No – Tools report was not documented and/or the full report is not readily available.

4.5.1. How to improve this score

If a report is provided with all prerequisite parts (which tools the protocol used, how they were configured, which results and what was done about the results), this metric receives a "Yes".

4.6. Q12 - Were the smart contracts deployed to a testnet? (Y/N)

Scoring: Y (Y/N)

Notes.LandX uses Rinkeby, as per its Build and Test page: https://landxit.github.io/land-x-smart-contracts/#build-and-test . However, Rinkeby testnet deployment evidence is not documented.

Guidance:

Yes - Protocol has proved their tesnet usage by providing the addresses

No - Protocol has not proved their testnet usage by providing the addresses

4.6.1. Oracles

The score for this section is derived from the individual scores of the following questions:

7) Are Oracles applicable to this protocol? (Y/N)
8) Is the Oracle sufficiently documented? (%)
9) Is front running mitigated by this protocol? (Y/N)
10) Can flashloan attacks be applied to the protocol, and if so, are those flashloan attack risks mitigated? (Y/N)

4.7. Q13 – Are Oracles applicable to the protocol? (Y/N)

If Oracles are applicable, Answer Yes to this question, Q14 to Q16 are counted in the score

Scoring: Y (Y/N)

4.8. 14 - Is the protocol's Oracle sufficiently documented? (%)

Scoring: 100 (%)

Notes: The oracle source is identified as Chainlink here. The oracles' timeframe and affected contracts are also mentioned.

Guidance:

100% - If it uses one, the Oracle is specified. The contracts dependent on the oracle are identified.

Basic software functions are identified (if the protocol provides its own price feed data).
Timeframe of price feeds are identified.

Or

100% - The reason as to why the protocol does not use an Oracle is identified and explained.
75% - The Oracle documentation identifies both source and timeframe, but does not provide additional context regarding smart contracts.
50% - Only the Oracle source is identified.
0%: No oracle is named / no oracle information is documented.

4.9. 15 - Is front running mitigated by this protocol? (Y/N)

Scoring: Y (Y/N)

Notes Front running mitigation techniques are described for LandX here:
https://landxit.github.io/land-x-smart-contracts/#frontrunning-flashloans

Guidance:
Yes - The protocol cannot be front run and there is an explanation as to why OR documented front running countermeasures are implemented.
No - The protocol does not mention front running or does not document any countermeasure against it

4.10. 16 - Can flashloan attacks be applied to the protocol, and if so, are those flashloan attack risks mitigated? (Y/N)

Scoring: Y (Y/N)

Notes Flashloan attack mitigation techniques are described for LandX here:
https://landxit.github.io/land-x-smart-contracts/#frontrunning-flashloans

"cTokens are not expected to trade on external exchanges and pricing is not affected by trade volume which mitigates the risk of a flashloan attack."

Guidance:
Yes: Flashloan attack countermeasures have been implemented and documented.
No: No flashloan attack countermeasure is documented.