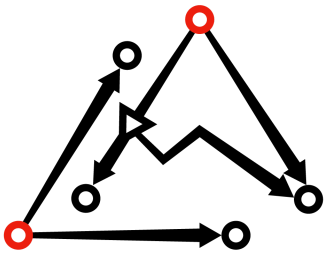

Least Authority

PRIVACY MATTERS

Audit Preparation Checklist

The following audit preparation checklist outlines the requirements a project should consider prior to undergoing an external security audit. Preparation for a security audit contributes both to the success and efficiency of the security review, in addition to the overall security of the system being assessed for exploits and vulnerabilities.

Audit Preparation Checklist	Optimal	Minimal
1. Create and Maintain Up-To-Date Project Documentation	✔	
Developer Documentation	✔	✔
Code Comments	✔	
Design Specification*	✔	✔
User Documentation	✔	
2. Reduce Unnecessary Complexity and Simplify the Code	✔	
3. Implement a Test Suite	✔	✔
4. Manage and Maintain Dependencies	✔	
5. Identify a Stable Audit Target in the Development Roadmap	✔	✔

*This is a minimal requirement for complex and rigorous systems.

1. **Create and Maintain Up-To-Date Project Documentation.**

    a. **Developer Documentation**: General documentation providing a high-level description of the system, each of the components, and interactions between those components. This can include developer documentation, new developer onboarding documentation, and architectural diagrams. This allows an auditing team to assess the in-scope components and understand the expected behavior of the system being audited.

    a. **Code Comments**: The documentation contained within the code should be comprehensive and explain the intended functionality of each of the components (e.g. functions or entry points).

    b. **Design Specification**: A design specification provides detailed and concise information about the system design and requirements. A design specification allows security auditors to check whether the code has been implemented correctly and adheres to the specification, and avoids incorrect assumptions about the expected behavior of the system, which may lead to missed security vulnerabilities.

    c. **User Documentation**: Comprehensive user documentation helps to ensure users interact with the system correctly and as intended, which encourages secure and correct usage.

2. **Reduce Unnecessary Complexity and Simplify the Code.** Increased complexity results in an increased attack surface, difficulty in understanding the code by security auditors, and difficulty maintaining the code by developers. It is recommended to reduce complexity and only implement features required by the system design in order to achieve the intended functionality.

3. **Implement a Test Suite.** Sufficient test coverage for success and failure cases, which helps to identify potential edge cases, and helps protect against errors and bugs, which may lead to vulnerabilities or exploits. A test suite should include a minimum of unit tests and integration tests. End-to-end testing is also recommended so that it can be determined if the implementation behaves as intended.

    A script should be provided with clear instructions so that the tests can be run via a single command.

4. **Manage and Maintain Dependencies**. Out of date, unmaintained, and unaudited dependencies increase the attack surface of any system. Use of audited and well maintained dependencies prior to a security audit allows security auditors to focus their review efforts on the system's behavior and the potential for security exploits.

5. **Identify a Stable Audit Target in the Development Roadmap.** A security audit should take place at a milestone in the development roadmap where the current version of the system has been completed and no major changes to the implementation are expected. Auditing code that is actively being developed may result in missed security issues and an inefficient security review. A security audit should be conducted on a single stable Git commit of the coded implementation.