HACKEN

2026 Smart Contract Audit Readiness Checklist

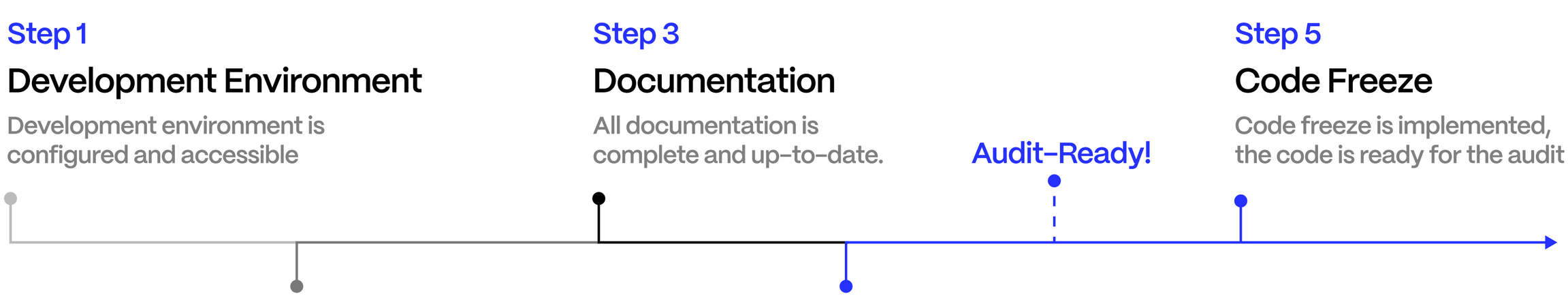
hacken.io

2026 Smart Contract Audit Readiness Checklist

Applicable for Smart Contracts, Protocols, Web3 & Web2 Projects

This pre-audit checklist helps teams prepare for a security assessment, ensuring efficient reviews, accurate findings, and high-quality results. Poor preparation can lead to delays, incomplete findings, and reduced audit accuracy.

Five Simple Steps Pre-Audit To to Get The Most Of Your Security Assessment



1. Development Environment

The project should have a development environment with a testing suite configured. It can be any development environment preferred by the customer development team.

Advantages:

- **Enhanced debugging:** Facilitates tracking down and fixing coding errors efficiently.
- **Simplified testing:** You can freely test and break things without affecting the live code.
- **Better version control:** Helps manage different versions of your code and track changes.
- **Reduced risk of accidental live deployment:** Avoids premature or accidental deployment of unfinished code.

Key requirements:

- [ ] No private dependencies should be included. All dependencies must be downloaded without additional configuration or manual setup.
- [ ] Any other global packages except a packages manager and a language compiler are required.
- [ ] The setup is OS-agnostic and can be run on Windows, Linux, or macOS.
- [ ] Run instructions are provided and, if executed, allow the successful compilation of the source code.

2. Clean and Executable Code

Key requirements:

- [ ] The code adheres to an official language style guide.
- [ ] The code compiles successfully without errors.
- [ ] All TODO, FIXME, and temporary debug comments are removed.
- [ ] Remove console.log / hardhat/console.sol imports
- [ ] Remove dead code and unused imports

3. Documentation

The project must include sufficient technical and functional documentation, available in English.

Technical documentation describes and explains anything related to the software product, ranging from internal documentation for teams to external documentation written for end users.

Key requirements for technical documentation:

- [ ] Programming languages and technologies utilized
- [ ] Instructions for deployment
- [ ] Usage of third party dependencies/programs
- [ ] Development environment description
- [ ] Run instructions
- [ ] Tests run instructions
- [ ] Benchmarks
- [ ] System architecture and internal/external interactions

Functional documentation define the intended system behavior and invariants, helping verify that the code truly meets project goals and user needs.

Key requirements for functional documentation:

- [ ] Functional requirements should be clear, specific, and easy to understand.
  - [x] Users should be able to earn tokens.
  - [x] The contract should enable users to stake ABC tokens and earn XYZ tokens as rewards.
- [ ] Functional requirements are testable so that they can be used to verify that the product is functioning correctly.
- [ ] End-user interaction flows are clearly defined.
- [ ] System inputs and outputs are explicitly specified.

- [ ] Behavioral constraints and limitations are documented.
- [ ] Performance and reliability requirements are stated.
- [ ] Configuration and deployment parameters

  Configurable parameters and their valid ranges
  - Minimum and maximum values for numerical inputs
  - Token decimal precision assumptions
  - Time-based constraints (lock periods, vesting intervals, voting periods, rate limits, caps)
- [ ] Access Control Matrix
  - Role definitions and their permissions
  - Function-level access restrictions
  - Privilege escalation paths (intended and restricted)
  - Multi-sig requirements and thresholds
- [ ] Upgrade/migration procedures

4. Prepare Testing Suite and Unit Tests

Unit tests can uncover security issues before an audit and significantly reduce audit time and cost by allowing auditors to focus on higher-risk areas.

Advantages:

- **For customers:** Validate that core functionality works as intended and help prevent costly bugs or fund losses.
- **For auditors:** Provide a clear baseline to verify system behavior and speed up validation of identified issues.

Recommendations to follow:

- [ ] Cover positive cases (happy paths)
- [ ] Cover negative cases
- [ ] Cover cases of construct usage by multiple users
- [ ] Ensure that code coverage plugin is configured
- [ ] Ensure that any additional required configuration (e.g. etherscan keys) is documented.

5. Code Freeze

A code freeze is the practice of halting changes to the codebase during an audit to ensure auditors review a stable and consistent version, improving the reliability of findings.

Advantages:

- **Stability:** Auditors work with an unchanging codebase.
- **Accuracy:** Minimizes errors caused by ongoing updates.
- **Efficiency:** Avoids delays from reviewing newly introduced changes.

Example: Implementing Code Freeze with GitHub

1. **Create a Release Branch**:
   A release branch, such as `release-audit`, should be created from the main branch to isolate the version being audited.

```
git checkout -b release-audit
git push origin release-audit
```

2. **Restrict Changes:** Use GitHub's branch protection rules to prevent updates to the `release-audit` branch. Navigate to the repository settings and:
   - Access Settings > Branches > Add Rule.
   - Select the `release-audit` branch.
   - Enable "Require pull request reviews before merging" and "Restrict who can push to matching branches."
3. **Enforce Freeze:** Announce the code freeze to the team, specifying that non-critical updates should be committed to other branches and merged later.
4. **Manage Fixes:** If any vulnerability arises during the audit, apply the necessary changes in the `release-audit` branch and notify the auditors of the updates.

Hacken is a leader in blockchain security, combining deep technical expertise in Web3 with real-time data-driven insights.

6800+
vulnerabilities found

$430B
on-chain assets verified

1B+
transactions monitored

$15M
paid out in bug bounties

50+
centralized exchanges

30K+
malicious contracts detected

60+
certified security engineers

ISO 27001
certified

Evolving alongside the industry for 8+ years

European Commission · VARA · European Blockchain Sandbox · Enterprise Ethereum Alliance · INATBA · Dubai Blockchain Center · C4 · The Digital Chamber · ERC3643 · Crypto Valley · مركز قطر للمال Qatar Financial Centre · The Millennium Alliance

Trusted by 1500+ digital asset leaders

MetaMask · ethereum foundation · ebsi European Blockchain · ADGM · FORGE Societe Generale Group · Binance · Bybit · crypto.com · OKX · KuCoin · Chainlink · Mawari · nubank · Aurora · Sui

Our Story

Unlike traditional providers, Hacken was born on blockchain, combining deep technical expertise in Web3 with real-time data-driven insights. Today Hacken is a leader in cybersecurity, trusted by enterprises, startups, and regulators to secure the new digital asset economy.

Learn more

hacken.io

Follow us on social media

linkedIn X