**Sudo Pallet Centralization Risk (Acknowledged)**Major (Centralization/Privilege)

In the Xode runtime, the sudo account has the authority to execute dispatchable functions that require the Root privileges including but not limited to calling the **set\_code()** function, used to upgrade the runtime code.

Any compromise to the sudo account may allow a hacker to take advantage of this authority and control the chain's runtime and introduce/execute potential malicious functionality in the runtime.

Here are some feasible short-term and long-term suggestions that would mitigate the potential risk to a different level and suggestions that would permanently fully resolve the risk.

**Short Term**

A combination of a time-delayed proxy and a multi-signature (⅔, ⅗) wallet mitigates the risk by delaying the sensitive operation and avoiding a single point of key management failure. This includes:

* A time-delayed proxy with reasonable latency, such as 48 hours, for community awareness of privileged operations; AND
* Assignment of privileged roles to multi-signature wallets to prevent a single point of failure due to a private key compromise; AND
* A medium/blog link for sharing the time-lock contract and multi-signers addresses information with the community.

For remediation and mitigated status, please provide the following information:

* Provide the time-delayed proxy configuration.
* Provide the multi-signature account configuration.
* Provide a link to the medium/blog with all of the above information included

**Long Term**

A combination of a time-delayed proxy on the contract upgrade operation and a DAO for controlling the upgrade operation mitigates the contract upgrade risk by applying transparency and decentralization.

* A time-delayed proxy with reasonable latency, such as 48 hours, for community awareness of privileged operations; AND
* Introduction of a DAO, governance, or voting pallet to increase decentralization, transparency, and user involvement; AND
* A medium/blog link for sharing the time-lock contract, multi-signer addresses, and DAO information with the community.

For remediation and mitigated status, please provide the following information:

* Provide the time-delayed proxy configuration.
* Provide the implementation of the DAO used.
* Provide a link to medium/blog with all the above information included.

**Permanent**

* Removing the pallet-sudo and/or renouncing the sudo account can fully resolve the risk.
* Renounce the account and never assign back the privileged role

The finding status will be updated once the centralization mitigations are applied on the running system and the respective account addresses are configured.

Below are the following code audit findings that have been resolved.

**1.** **Randomness Implementation in Contracts Pallet (Informational)**

* The contract pallet requires a Random trait implementation in its runtime configuration. The code in scope provides a mock implementation that constantly returns a constant value.
* The audit team requests clarification on such behavior, if it is an expected one, and whether dedicated documentation will be released about contract development in the Xode ecosystem.

**2.** **Default Developer Template in Production Codebase (Informational)**

* The Xode runtime includes FRAME pallets and a template pallet from the Polkadot-SDK repository, which provides only boilerplate code with extrinsic like do something, do another thing, and cause error, lacking meaningful functionality.

**3.** **Missing Documentation on Contracts Balance Dispatchable (Informational)**

* This custom type enables contracts to call balance transfer functions from the Balance pallet. Its behavior is unclear due to the lack of documentation, making it crucial to clarify its role in the code for better understanding and awareness.

**4.** **Library Trait Redefinition from Updated SDK Version (Informational)**

* The DenyThenTry and DenyReserveTransferToRelayChain traits are included in the xcm module in the latest Polkadot SDK version, so they can be imported and used instead of being directly defined in the runtime.
* **Note:** They recommended to use the versions from the XCM crate to improve code clarity and modularity.

**5.** **Missing Mainnet Configuration in (Informational & Logical Issue) –** chain\_spec

* The DenyThenTry and DenyReserveTransferToRelayChain traits are included in the xcm module in the latest Polkadot SDK version, so they can be imported and used instead of being directly defined in the runtime.
* **Note:** They recommended to use the versions from the XCM crate to improve code clarity and modularity.

**6.** **General Code Optimization (Optimization & Coding Style)**

* Rust is a powerful and extensible language, offering many tools and patterns for efficient implementation. However, its wide range of possibilities can sometimes lead to code lacking optimizations or common patterns. The Clippy tool helps improve this by providing suggestions for better readability and efficiency.
* **Note:** They recommend running the Clippy tool and following the suggestions it returns.

**Note:** The last audit was delivered on 08/07/2024

The auditing process pays special attention to the following considerations:

* Testing the smart contracts against both common and uncommon attack vectors.
* Assessing the codebase to ensure compliance with current best practices and industry standards.
* Ensuring contract logic meets the specifications and intentions of the client.
* Cross referencing contract structure and implementation against similar smart contracts produced by industry
* leaders.
* Thorough line-by-line manual review of the entire codebase by industry experts.

The security assessment resulted in findings that ranged from critical to informational. We recommend addressing these findings to ensure a high level of security standards and industry practices. We suggest recommendations that could better serve the project from the security perspective:

* Testing the smart contracts against both common and uncommon attack vectors;
* Enhance general coding practices for better structures of source codes;
* Add enough unit tests to cover the possible use cases;
* Provide more comments per each function for readability, especially contracts that are verified in public;
* Provide more transparency on privileged activities once the protocol is live.

**Out-of-scope dependencies**

The Xode Blockchain, built on the Polkadot SDK, extensively uses third-party modules such as standard FRAME pallets.

Additionally, the project utilizes various libraries that might have vulnerabilities in specific versions or be deprecated. It's recommended to always update these dependencies to the latest stable version and monitor for new vulnerabilities using tools like cargo audit.

These third-party modules, along with custom pallets outside the audit's scope, were treated as black boxes, assuming their functional correctness. The detailed scope of the audited Xode repository is outlined in the report's scope section.

**Testing and Documentation**

The project does not include tests to cover the codebase by running specific scenarios, including both successful and reproduction of error conditions. Moreover, more complex scenarios involving multiple users and multiple contract interactions (e.g., multiple deposits on different epochs) are recommended in order to consider realistic cases emulating the real environment in which the contracts will be deployed.

The project also includes comments in the code to explain most of the functionalities or the rationale behind some assumptions taken. However, a detailed documentation of the design choices made and an explanation of the implementation details with the rationale behind it is missing or incomplete. It is strongly recommended to produce clear and detailed documentation to increase the maintainability of the project and allow the community to easily understand and, therefore, trust in the project.

**Weights**

In the Polkadot SDK framework, weights are used to manage the time it takes to validate a block. By controlling the execution time that a block can consume, weights set limits on storage input, output, and computation. Weights are configured on extrinsics, i.e. calls coming from users. However, since misconfigured weights could lead to a potential Denial-of-Service (DoS), we invite the Xode team to carefully select the weights and to always keep the benchmarking process updated to compute the optimal weights at all times.

**Development and Testnet Configuration**

The current codebase includes chain configurations for a potential development network and a testnet. There is not any reference to a production chain specification. We recommend reviewing the deployment and genesis parameters of the future production chain. Specifically, the parameters impacting the size and depth of the contract execution in the pallet\_contract have been modified to accommodate development requirements that may not be a good fit for the public and potential adversarial environment.