## Audit Report

# Polytone

v1.0

June 5, 2023

# Table of Contents

# License

# Disclaimer



This audit has been performed by

**Oak Security**

https://oaksecurity.io/
info@oaksecurity.io

# Introduction

## Purpose of This Report

Oak Security has been engaged by In With The New to perform a security audit of the Polytone CosmWasm smart contracts.

The objectives of the audit are as follows:

1. Determine the correct functioning of the protocol, in accordance with the project specification.
2. Determine possible vulnerabilities, which could be exploited by an attacker.
3. Determine smart contract bugs, which might lead to unexpected behavior.
4. Analyze whether best practices have been applied during development.
5. Make recommendations to improve code safety and readability.

This report represents a summary of the findings.

As with any code audit, there is a limit to which vulnerabilities can be found, and unexpected execution paths may still be possible. The author of this report does not guarantee complete coverage (see disclaimer).

## Codebase Submitted for the Audit

The audit has been performed on the following target:

| | |
|---|---|
| Repository | https://github.com/DA0-DA0/polytone |
| Commit | `9733a273ecf84c46c9e51e2bdc2ed00c8347f9cd` |
| Scope | All contracts were in scope. |

# Methodology

The audit has been performed in the following steps:

1. Gaining an understanding of the code base's intended purpose by reading the available documentation.
2. Automated source code and dependency analysis.
3. Manual line-by-line analysis of the source code for security vulnerabilities and use of best practice guidelines, including but not limited to:
    a. Race condition analysis
    b. Under-/overflow issues
    c. Key management vulnerabilities
4. Report preparation

# Functionality Overview

Polytone enables users on chain A to execute messages and perform queries on a remote chain B through IBC.
The protocol relies on the `note` contract on chain A that connects and controls a `voice` contract on chain B. Each user from chain A executes messages and queries through their personal `proxy` contract.

# How to Read This Report

This report classifies the issues found into the following severity categories:

| Severity | Description |
|---|---|
| **Critical** | A serious and exploitable vulnerability that can lead to loss of funds, unrecoverable locked funds, or catastrophic denial of service. |
| **Major** | A vulnerability or bug that can affect the correct functioning of the system, lead to incorrect states or denial of service. |
| **Minor** | A violation of common best practices or incorrect usage of primitives, which may not currently have a major impact on security, but may do so in the future or introduce inefficiencies. |
| **Informational** | Comments and recommendations of design decisions or potential optimizations, that are not relevant to security. Their application may improve aspects, such as user experience or readability, but is not strictly necessary. This category may also include opinionated recommendations that the project team might not share. |

The status of an issue can be one of the following: **Pending,** **Acknowledged**, or **Resolved**.

Note that audits are an important step to improving the security of smart contracts and can find many issues. However, auditing complex codebases has its limits and a remaining risk is present (see disclaimer).

Users of the system should exercise caution. In order to help with the evaluation of the remaining risk, we provide a measure of the following key indicators: **code complexity**, **code readability**, **level of documentation**, and **test coverage**. We include a table with these criteria below.

Note that high complexity or low test coverage does not necessarily equate to a higher risk, although certain bugs are more easily detected in unit testing than in a security audit and vice versa.

# Code Quality Criteria

The auditor team assesses the codebase's code quality criteria as follows:

| Criteria | Status | Comment |
|---|---|---|
| Code complexity | Medium | - |
| Code readability and clarity | Medium-High | - |
| Level of documentation | Medium | - |
| Test coverage | Medium | While unit test coverage is low with 22.22% according to cargo tarpaulin, and some integration tests are failing, we classify the overall test coverage as medium due to the extensive simulation test suite. |

# Summary of Findings

| No | Description | Severity | Status |
|---|---|---|---|
| 1 | Sequence is not reset when establishing a new channel | Major | Resolved |
| 2 | Controllers could instantiate proxies and execute messages on the receiver chain on behalf of other users without their consent | Major | Resolved |
| 3 | Encoding extensions are not enforced during handshake | Minor | Acknowledged |
| 4 | The proxy contract does not return the correct index of an errored message | Minor | Resolved |
| 5 | `PROXY_CODE_ID` and `BLOCK_MAX_GAS` are not validated | Minor | Resolved |
| 6 | The note contract cannot handle more than `2^64-1` messages | Minor | Acknowledged |
| 7 | “Migrate only if newer” pattern not followed | Informational | Acknowledged |
| 8 | Typographic errors affect readability | Informational | Resolved |
| 9 | Nondescript storage keys may make storage state difficult to maintain | Informational | Acknowledged |
| 10 | Events are not always emitted | Informational | Resolved |
| 11 | Overflow checks not enabled for release profile | Informational | Resolved |
| 12 | Usage of `unwrap` and `expect` goes against best practices | Informational | Acknowledged |
| 13 | Return detailed error message rather than generic error | Informational | Resolved |
| 14 | `UnUnordered` error name may confuse users | Informational | Resolved |

# Detailed Findings

## 1. Sequence is not reset when establishing a new channel

**Severity: Major**

In `packages/polytone/src/callback.rs:90-117`, the `request_callback` function reads the latest value stored in the `SEQ` storage and saves it as the callback for the incoming packet sequence. However, the `SEQ` storage does not reset the sequence back to `1` when the old channel is closed via `ibc_channel_close`.

Suppose the old channel is closed and a new channel is established. The packet sequence will start as `1`. Since the `SEQ` storage records the old channel's sequence value, the `CALLBACKS` storage will save the pending callback with an incorrect `seq` value.

As a result, pending callbacks will never be executed properly by the intended receiver, causing `CALLBACKS` not to be handled and `LOCAL_TO_REMOTE_ACCOUNT` not to save the request initializer's remote proxy contract address.

### Recommendation

We recommend resetting the `SEQ` storage when closing the channel.

**Status: Resolved**

## 2. Controllers could instantiate proxies and execute messages on the receiver chain on behalf of other users without their consent

**Severity: Major**

The `on_behalf_of` attribute of the `Execute` struct defined in `contracts/note/src/msg.rs:61` allows controlled `note` contracts to execute messages on the receiver chain on behalf of other users.

However, this is possible without the targeted user's consent allowing the controller account to impersonate unaware users on the controlled chains.

While this cannot directly affect targeted users and their funds, it could allow a malicious controller to perform phishing and fraud campaigns or impersonate users in order to compromise their reputation.

### Recommendation

We recommend requiring the user's consent before relaying messages "on behalf" to the controlled chain in the `note` contract. For example, a byte array signed by the `on_behalf_of` account could be required.

**Status: Resolved**

## 3. Encoding extensions are not enforced during handshake

**Severity: Minor**

The Polytone protocol defines encoding extensions in order to ensure that connected chains exchange data with an agreed format.

To do so, in `packages/polytone/src/handshake/note.rs:35`, the protocol requires that the `voice` contract's extensions are a subset of the `note` contract's.

However, the implementation, instead of exchanging the underlying chain-supported encodings, exchanges the hardcoded "`JSON-CosmosMsg`" string defined in `contracts/note/src/ibc.rs:26` and `41`.

This would cause the mentioned validation check during the handshake to succeed without actually enforcing the extensions subset requirement.

### Recommendation

We recommend substituting the hardcoded "`JSON-CosmosMsg`" string with the chain-supported encoding extensions.

**Status: Acknowledged**

## 4. The proxy contract does not return the correct index of an errored message

**Severity: Minor**

In `contracts/proxy/src/contract.rs:78`, the `proxy` contract's `reply` handler returns an error containing the index of the errored message.

However, it returns the cardinality of submitted messages instead of the actual index. This would cause the propagation of an error with incorrect information leading to potentially wrong actions in the callback.

### Recommendation

We recommend substituting the `index` field of the error with `msg.id` instead of `collector.len()`.

**Status: Resolved**

## 5. `PROXY_CODE_ID` and `BLOCK_MAX_GAS` are not validated

**Severity: Minor**

When instantiating the `note` and `voice` contracts, no validations are performed to validate the `PROXY_CODE_ID` and `BLOCK_MAX_GAS` values. This is problematic because misconfiguring the `PROXY_CODE_ID` to zero would cause the proxy instantiation to fail in `contracts/voice/src/contract.rs:93`.

As for `BLOCK_MAX_GAS` value, it should be validated to be higher than `ACK_GAS_NEEDED` and `ERR_GAS_NEEDED` to prevent underflows in `contracts/voice/src/ibc.rs:102` and `contracts/note/src/ibc.rs:102` and `123`.

We classify this issue as minor because the contract admin can use the `MigrateMsg::WithUpdate` message to recover from these situations.

### Recommendation

We recommend validating the `PROXY_CODE_ID` and `BLOCK_MAX_GAS` values during instantiation and migration.

**Status: Resolved**

## 6. The note contract cannot handle more than `2^64-1` messages

**Severity: Minor**

In `packages/polytone/src/callback.rs:97`, the message sequence `SEQ` is incremented and saved.

However, since it is stored as a `u64`, the increment will overflow, and panic because of the `overflow-check` in place in the `Cargo.toml`, after the execution of `2^64-1` messages.

This would prevent users from interacting with the contract and consequently lose access to their proxy accounts in the controlled chain.

### Recommendation

We recommend handling `SEQ` as a ring counter.

**Status: Acknowledged**

## 7. "Migrate only if newer" pattern not followed

**Severity: Informational**

The contracts within the scope of this audit are currently migrated without regard to their version. This can be improved by adding validation to ensure that the migration is only performed if the supplied version is newer.

### Recommendation

We recommend following the migrate "only if newer" pattern defined in the CosmWasm documentation.

**Status: Acknowledged**

## 8. Typographic errors affect readability

**Severity: Informational**

Several typographical errors were found during the audit:

- `contracts/note/src/msg.rs`
  - In line `18`, the word `mat` should be `may`
  - In line `26`, the word `initators` should be `initiator's`
  - In line `59`, the word `optionaly` should be `optionally`
- `contracts/voice/src/ibc.rs`
  - In line `32`, the word `dererministicly` should be `deterministically`
- `packages/polytone/src/callback.rs`
  - In line `22`, the word `Initator` should be `Initiator`
  - In line `193`, the word `Disembiguates` should be `Disambiguates`
- `packages/polytone/src/ack.rs`
  - In line `27`, the word `succeded` should be `succeeded`
  - In line `52`, the word `unmarshaling` should be `unmarshalling`
- `packages/polytone/src/handshake/error.rs`
  - In line `6`, the word `ProtocolMissmatch` should be `ProtocolMismatch`

This affects the readability of the contracts.

### Recommendation

We recommend correcting the typographic errors as described above above.

**Status: Resolved**

## 9. Nondescript storage keys may make storage state difficult to maintain

**Severity: Informational**

Multiple storage entries defined in `contracts/note/src/state.rs:5-16` use a single letter as their storage key. This contradicts best programming practices as the keys are not descriptive, and could lead to naming conflicts in the future. Additionally, developers might find it hard to introduce new states or to group similar values together in the future, as that may contradict the current implementation of using the next character in the sequence.

### Recommendation

We recommend using more descriptive storage keys.

**Status: Acknowledged**

## 10. Events are not always emitted

**Severity: Informational**

There are multiple functions within the scope of this audit that do not emit events or attributes. It is best practice to emit events and attributes to improve the usability of the contracts and to support off-chain event listeners and blockchain indexers.

The following functions do not emit events or attributes:

- In `contracts/note/src/contract.rs`, the `instantiate` and `migrate` functions should emit an event containing the `BLOCK_MAX_GAS` value.
- In `contracts/voice/src/contract.rs`, the `instantiate` and `migrate` functions should emit an event containing the `PROXY_CODE_ID` and `BLOCK_MAX_GAS` values.

### Recommendation

We recommend emitting events with all the attributes involved in the computation.

**Status: Resolved**

## 11. Overflow checks not enabled for release profile

**Severity: Informational**

The following packages and contracts do not enable `overflow-checks` for the release profile:

- `contracts/note/cargo.toml`
- `contracts/voice/cargo.toml`
- `contracts/proxy/cargo.toml`
- `packages/polytone/cargo.toml`

While enabled implicitly through the workspace manifest, a future refactoring might break this assumption.

### Recommendation

We recommend enabling overflow checks in all packages, including those that do not currently perform calculations, to prevent unintended consequences if changes are added in future releases or during refactoring. Note that enabling overflow checks in packages other than the workspace manifest will lead to compiler warnings.

**Status: Resolved**

## 12. Usage of `unwrap` and `expect` goes against best practices

**Severity: Informational**

In `contracts/proxy/src/contract.rs:88` and `contracts/voice/src/ibc.rs:117`, `unwrap` and `expect` are used. Usage of these functions is generally discouraged because they lead to panics without developer-friendly error messages.

Panics also cause the wasm execution to abort rather than unwind, which does not allow handling of the error from the calling context.

### Recommendation

We recommend handling errors more gracefully instead of letting the code panic.

**Status: Acknowledged**

### 13. Return detailed error message rather than generic error

**Severity: Informational**

In `contracts/note/src/contact.rs:98`, a generic error is returned after checking if the note is controlled. It is best practice to return a detailed error message as done in line `95`.

#### Recommendation

We recommend returning a detailed error message as done in line `95`.

**Status: Resolved**

### 14. `UnUnordered` error name may confuse users

**Severity: Informational**

The custom error in `packages/polytone/src/handshake/error.rs:8` is currently named `UnUnordered`. The double negation in its name might confuse users.

#### Recommendation

We recommend using a less confusing name for the error message, such as `ExpectUnordered`.

**Status: Resolved**

# Threat Model Analysis

## Methodology

### Process Applied

The process performed to analyze the system for potential threats and build a comprehensive model is based on the approach first pioneered by Microsoft in 1999 that has developed into the STRIDE model (https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20). Whilst STRIDE is aimed at traditional software systems, it is generic enough to provide a threat classification suitable for blockchain applications with little adaptation (see below).
The result of the STRIDE classification has then been applied to a risk management matrix with simple countermeasures and mitigations suitable for blockchain applications.

### STRIDE Interpretation in the Blockchain Context

STRIDE was first designed for closed software applications in permissioned environments with limited network capabilities. However, the classification provided can be adapted to blockchain systems with small adaptations. The below table highlights a blockchain-centric interpretation of the STRIDE classification:

| | |
|---|---|
| **Spoofing** | In a blockchain context, the authenticity of communications is built into the underlying cryptographic public key infrastructure. However, spoofing attack vectors can occur at the off-chain level and within a social engineering paradigm. An example of the former is a Sybil attack where an actor uses multiple cryptographic entities to manipulate a system (wash-trading, auction smart contract manipulation, etc.).<br>The latter usually consists of attackers imitating well-known actors, for instance, the creation of an impersonation token smart contract with a malicious implementation. |
| **Tampering** | Similarly to spoofing, tampering of data is usually not directly relevant to blockchain data itself due to cryptographic integrity. One example of this is a supply chain attack that manages to inject malicious code or substitute trusted software that interacts with the blockchain (node software, wallets, libraries). |
| **Repudiation** | Repudiation, i.e. the ability of an actor to deny that they have taken action is usually not relevant at the transaction level of blockchains. However, it makes sense to maintain this category, since it may apply to |

| | |
|---|---|
| | additional software used in blockchain applications, such as user-facing web services. An example is the claim of a loss of a private key and hence assets. |
| **Information Disclosure** | Information disclosure has to be treated differently at the blockchain layer and the off-chain layer. Since the blockchain state is inherently public in most systems, information leakage here relates to data that is discoverable on the blockchain, even if it should be protected. Predictable random number generation could be classified as such, in addition to simply storing private data on the blockchain. In some cases, information in the mempool (pending/unconfirmed transactions) can be exploited in front-running or sandwich attacks.<br>At the off-chain layer, the leakage of private keys is a good example of operational threat vectors. |
| **Denial of Service** | Denial of service threat vectors translates directly to blockchain systems at the infrastructure level.<br>At the smart contract or protocol layer, there are more subtle DoS threats, such as unbounded iterations over data structures that could be exploited to make certain transactions not executable. |
| **Elevated Privileges** | Elevated privilege attack vectors directly translate to blockchain services. Faulty authorization at the smart contract level is only one example where users might obtain access to functionality that should not be accessible. |

# Assets

The following describes assets that may be valuable to an attacker or other stakeholders of the system.

## Cosmos messages

Data sent to other blockchains via Inter-Blockchain Communication (IBC) protocol or similar cross-chain communication methods. An attacker finds the integrity of these communications valuable since they could transport data for executing valuable actions on the other chain, such as transferring assets, triggering smart contract functions, or interacting with decentralized applications.

## Fungible tokens

Tokens of any denomination (e.g., JUNO, ATOM, or OSMO) of type `sdk.Coin`. These tokens hold value as they can be traded on a market such as decentralized exchanges for profit, or used for governance or to extract value in various ways, making them attractive targets for attackers.

## Non-fungible tokens

NFTs represent unique digital assets and associated rights such as art, music, or media with inherent scarcity. NFTs can be sold on a market such as decentralized exchanges for profit or used to extract value in other ways, making them attractive targets for attackers.

## Query responses

Information queried from other blockchains is valuable to attackers, as other protocols might rely on the accuracy and integrity of the query response.

## Accounts

An attacker finds access to ICA accounts on the controlled blockchain valuable since that allows control of tokens held in or privileges associated with these accounts. Additionally, access to smart contract accounts allows attackers to control the smart contract and perform malicious activities.

## Private keys controlling external accounts

In the case of externally owned accounts, some entity controls a private key in control of that account. An attacker finds access to a private key valuable since that gives them full control over the account, allowing them to impersonate and act on behalf of the account owner.

## Migration authorization

In CosmWasm smart contracts, if there is a defined contract admin, the contract is not immutable. Gaining unauthorized access to act as the contract admin is valuable to an attacker, as it allows them to modify the contract's source code, potentially introducing backdoors or other malicious code.

## Liveness of the protocol

An attacker may find it valuable to target the liveness of the protocol, such as preventing users or the controller from sending Cosmos messages. This could allow the attacker to front-run unsent messages or disrupt the normal functioning of the system, for example delaying governance actions or in the case of DeFi protocols liquidations.

## Liveness of the underlying blockchains and relayers

Attacking the liveness of underlying blockchains and IBC relayers can also be valuable to an attacker. By doing so, they may profit from shorting the native tokens or extorting stakeholders to stop the attack, causing disruptions and instability in the blockchain ecosystem.

# Stakeholders/Potential Threat Actors

## Users

Main users of the protocol send `Execute` and `Query` messages. Users may try to impersonate other users or repudiate their actions, aiming to gain unauthorized access to resources or avoid accountability.

## Controllers

Privileged accounts that control `note` contracts. Controllers may attempt to impersonate users to gain access to their resources or manipulate the system in their favor.

## Contract admins

Contract admins have the power to migrate contracts. They may try to migrate contracts to a malicious codebase, introducing vulnerabilities or backdoors in the system.

## Validators

Nodes are responsible for verifying and validating transactions and adding them to the blockchain to produce blocks. Validators may get involved in blocking or censoring transactions, disrupting the normal functioning of the system. They can also re-order transactions and hence messages, potentially running front-/back-running or sandwich attacks.

## Relayers

Off-chain programs, called relayers, monitor for updates on open channels between sets of IBC-enabled chains and relay messages from one chain to the other. Since blockchains do not directly pass messages to each other over the network, relayers are a crucial part of the IBC infrastructure. However, if there is a single entity or consensus among colluding relayer operators, they may block or censor a particular message to be relayed to a counterparty blockchain.

## Governance participants

Governance participants are involved in the decision-making process related to the blockchain's governance, including creating and executing proposals. Depending on the enabled governance proposals, a successful governance vote could result in the ability to:

- Call sudo entry points in smart contracts, granting privileged access to specific functions or features within the contract.
- Perform actions on behalf of any users to any smart contracts, which might include modifying the contract’s owner or transferring funds from the contract.
- Perform a contract migration to modify its source code, potentially including a backdoor to the contract.

# Threat Model

## STRIDE Classification

The following threat vectors have been identified using the STRIDE classification, grouped by components of the system.

| | **Spoofing** | **Tampering** | **Repudiation** | **Information Disclosure** | **Denial of Service** | **Elevated Privileges** |
|---|---|---|---|---|---|---|
| **Externally owned account** | Lost account | Pharming phishing social engineering | Compromised account | Private key leakage<br><br>Doxxing/identity disclosure | DOS of infrastructure | Compromised private key |
| **Cosmos infrastructure** | - | Backdoors | - | Spyware | Relayers halt<br><br>Message filtering | Governance actions |
| **Execute/Query messages handling** | Recipient replacement | Data manipulation | - | Front-running | Execution loops | Unauthorized on-behalf execution |

## Mitigation Matrix

The following mitigation matrix describes each of the threat vectors identified in the STRIDE classification above, assigning an impact and likelihood and suggesting countermeasures and mitigation strategies. Countermeasures can be taken to identify and react to a threat, while mitigation strategies prevent a threat or reduce its impact or likelihood.

Externally owned account

| Threat Vector | Impact | Likelihood | Mitigation | Countermeasures |
|---|---|---|---|---|
| Lost account: The attacker claims that they own an account and the access to the private key has been lost | Medium | Low | Have a clear policy not to refund lost assets or restore privileges | Enforce policy and strictness |
| Pharming/phishing/social engineering:<br>The attacker may manipulate users' wallets, lure them to malicious front-end, manipulate DNS records, or use social engineering to trick users/teams into signing manipulated transactions transferring funds/permissions | Medium | Medium | Educate users and team, protect DNS records, create awareness, offer blacklists with malicious sites, create activity on social channels to build reputable channels, deploy front-ends on IPFS or other decentralized infrastructure | Monitor all systems, monitor communities and impersonations/malicious copies of official channels, communicate attempted pharming/phishing/social engineering, have processes in place to recover from DNS manipulation, attacks on front-ends quickly |
| Compromised account: The attacker claims they are a victim of scapegoating, denying responsibility for their attack | Low | Low | Have a clear policy not to refund lost assets or restore privileges | Enforce policy |

| | | | | |
|---|---|---|---|---|
| Private key leakage: Private keys are accidentally shared or logged | Medium | Medium | Educate users and team, ensure private keys are properly handled in wallet software, use hardware wallets/air-gapped devices, security keys, multi-signatures | - |
| Doxxing/identity disclosure: Private data such as the off-chain identity of users disclosed | Low | Medium | Educate users and team, no storage of identity/sensible data in databases that link identity to account addresses, follow privacy regulations and guidelines | - |
| DOS of infrastructure: DOS attack on the end user's device/network or on the blockchain node they interact with | Low | Low | Educate users and team, use firewalls, sentry architecture, load balancers, VPNs | Monitor infrastructure, and have processes in place to elastically provision and deploy additional resources |
| Compromised private key: Private keys may be compromised | Medium | Medium | Educate users and team | - |

Cosmos infrastructure

| Threat Vector | Impact | Likelihood | Mitigation | Countermeasures |
|---|---|---|---|---|
| Backdoors: Builder team/deployer may implement back doors and malicious code that modify Polytone messages before sending them to the other chain. | High | Low | Internal code reviews, unit testing, integration tests, automatic software engineering tools, audits | Monitor upgrade attempts, enforce processes on deployment, offer bug-bounties |
| Spyware: Builder team/deployer may manipulate code to share private keys or private information of users | Medium | Low | Internal code reviews, unit testing, integration tests, automatic software engineering tools audits | Monitor upgrade attempts, enforce processes on deployment, offer bug-bounties |
| Relayers halt: Relayers could stop relaying transactions to the recipient chain or discard processing transactions arbitrarily | High | Low | Run in-house and redundant relayers | Monitor the network |
| Message filtering: Underlying blockchains could filter polytone messages | High | Low | Internal code reviews, unit testing, integration tests, automatic software engineering tools, audits | - |
| Governance actions: Underlying blockchains could implement governance modules that enable governance to execute privileged operations on Polytone contracts, such as migrations. | High | Low | Educate users and the community about the project and its scope | Follow discussions and be active in the community |

Execute/Query messages handling

| Threat Vector | Impact | Likelihood | Mitigation | Countermeasures |
|---|---|---|---|---|
| Recipient replacement: The Polytone contracts could redirect messages/queries to an arbitrary recipient | High | Low | Internal reviews, audits, ensure process verifies that code has been audited | Enforce processes on deployment, offer bug-bounties |
| Data manipulation: The Polytone contracts could modify the query data before returning it or the message before broadcasting it | High | Low | Internal reviews, audits, ensure process verifies that code has been audited | Enforce processes on deployment, offer bug-bounties |
| Front-running: Messages are queued in the mempool and by the relayer before being executed in the controller chain and then broadcasted in the controlled chain leading to possible front-running | High | Medium | Run in-house and redundant relayers, deploy note contract on chain supporting encrypted mempools | - |
| Execution loops: Attackers could deploy another note contract in the controlled chain and execute a message targeting it from a note contract deployed in the controller chain. This would cause a loop of messages between the two chains targeting the relayer’s funds. | High | Medium | Implement a mechanism to track messages' origin and break loops. | Monitor the chain and stop the relayer if this attack occurs. |
| Unauthorized on-behalf execution: Execute messages and queries on behalf of another user without having authorization | High | Low | Internal reviews, audits, ensure process verifies that code has been audited | Enforce processes on deployment, offer bug-bounties |