Security Disclosure and Retrospective Analysis

Composable Foundation

Follow

--

Listen

Share

TL;DR

We recently identified and addressed an issue within our IBC pallet implementation. Specifically, we identified an incompatibility with the response of success acknowledgements coming from cross-chain ICS20 transactions, using ibc-hooks and packet forward middleware. Instead of recognizing these as successful transactions, our system erroneously treated them as failures, triggering an automatic (and erroneous) token refund.

Our team was quick to spot this discrepancy after just two transactions were executed. We immediately halted the relayer to prevent further issues. A comprehensive analysis was conducted, leading to a refund of 2363.23 PICA to the escrow account, while 0.0996 OSMO tokens in Picasso were burnt to correct the error.

Given the complexity of operating across multiple blockchain ecosystems, each with its own governance process, resolving this issue was a timely task. The fix itself was completed within approximately two hours, but the subsequent testing and implementation phases were more time-consuming as it required alignment and execution with all three chains involved. This resulted in a total approximate downtime of 14 days.

In accordance with best practices for vulnerability disclosure, such as those outlined in IBC-go's security guidelines, we refrained from publicizing this issue until we had notified all relevant parties, including the IBC-rs team. This was particularly important as there was a suspicion that a dependency could be the root cause of the issue.

Detailed explanation

On Wednesday 20th September 8:08 GMT we identified an issue when processing packet acknowledgements that were routed using TFM + ibc hooks to perform cross-chain swaps on Osmosis.

The ICS20 spec defines a FungibleTokenPacketSuccess as follows:

Upon receiving an acknowledgement packet, pallet-ibc will call the on-receive method.

As per ICS20 spec, when the ack-packet is not recognized as successful, pallet-ibc proceeds to refund the token. If it is the source chain it will un-escrow the tokens, or mint tokens if it is the destination chain.

However, in the case of ibc-hooks, the packet acknowledgement had a different "shape" (https://github.com/osmosis-labs/osmosis/blob/5779e89d813c70138503e0064692206f723c2422/x/ibc-hooks/types/types.go#L36). And this caused pallet-ibc to deem ack packets that were marked successful, incorrectly as error packets. This triggered an unintended refund process.

We identified two transactions/extrinzics where this occurred, and promptly paused our relayer to understand the root cause, contain the damage, and come up with a solution.

Offender 1

https://polkadot.js.org/apps/?rpc=wss%3A%2F%2Frpc.composablenodes.tech#/explorer/query/0xbabc174eee9c52b77ad353eea6803ee20edfe1977670d297c371c9ff9d2507eb

Here pallet-ibc refunded 0.099600 OSMO when it should not. We are burning 0.099600 OSMO to restore the supply balance in Picasso.

Offender 2

https://polkadot.js.org/apps/?rpc=wss%3A%2F%2Frpc.composablenodes.tech#/explorer/query/0x6e380e7744c084d8b74fbbc350b07cadd6f1eae02d41e96f41ddb6750ab5880c

Here, pallet-ibc refunded 2363.228225424771 PICA when it should not. We are sending 2363.228225424771 PICA to the escrow account to restore the supply balance on the bridge.

Referenda for burning 0.099600 OSMO for our Bridge fees account, and to add 2363.228225424771 PICA from the Bridge Fees account onto the Bridge escrow account.

https://picasso.polkassembly.io/referenda/4

Burn and Top up have been applied:

https://picasso.subscan.io/block/3369173?tab=event

Code fix

Fix on pallet-ibc: https://github.com/ComposableFi/composable-ibc/pull/420

Update on our Picasso + Composable Runtime: https://github.com/ComposableFi/composable/pull/4153

Become more defensive when reading the memo field: https://github.com/ComposableFi/composable/pull/4151

In keeping with the following security vulnerability, we have made our IBC implementation resist to reading unspecific memo fields: https://twitter.com/scvsecurity/status/1682329758020022272?s=46&t=47J2tBvbrsFNopirIM-O7g

Timeline:

September 20th

- 15:08 UTC: Identification of the issue
- 23:08 UTC: Maintenance page and announcement posted on app.trustless.zone

September 21st

- Fix identification

September 21st — 26th

- Ongoing fix, implementation, and testing

September 26th

- Chain upgrade: Picasso

October 2nd

- Chain upgrade: Composable Cosmos Chain (Centauri)

October 3rd

- Restart of the bridge
- Lifting of maintenance

The issue we encountered had both direct and indirect impact across the three chains we oversee — namely, Picasso, Composable, and our Composable Cosmos chain. As a result, we had to execute two runtime upgrades for Picasso and Composable, as well as a full chain upgrade for our Composable Cosmos chain.

Due to the intricate nature of the problem and the time required for comprehensive testing, the resolution process extended beyond the trust period, causing our light client to freeze. This necessitated the initiation of an additional governance proposal specifically aimed at upgrading the light client.

Key Insights, Future Directions, and Industry Recommendations

Enhanced Memo Field Validation

Our experience has underscored the necessity for robust validation mechanisms, particularly concerning the memo field. As a result, we have made our validation process stricter.

Importance of Monitoring

This incident highlighted the critical role of real-time monitoring. Our internal indexer alerted us to the issue, reinforcing its value in our operational stack. We are committed to investing further resources to enhance this monitoring capability, making it more robust and responsive.

Transition to Mainstream IBC Implementations

While our current IBC version is a fork of Informal System’s work, we recognize the importance of staying updated with developments. Transitioning to an upstreamed ibc-rs is imperative, as it allows us to benefit from a broader community of developers and experts scrutinizing the code.

Coordinated IBC Implementation

As the IBC ecosystem grows with various implementations by different projects like ourselves and other users of IBC-rs, there’s an increasing need for streamlined coordination. This is essential to prevent future issues related to disconnects or incompatibility. We view this as a significant contribution to the IBC community, extending beyond our own implementation efforts. By fostering better coordination, we can collectively ensure a more secure and interoperable landscape to bring IBC everywhere.

As contributors to IBC, these insights not only inform Composable’s immediate next steps but also contribute to the broader conversation around IBC implementation and extension. We are committed to sharing our learnings and collaborating with other projects to drive the IBC ecosystem forward.

Composable Change Management and Incident Response Process (Version 0.1)

In response to valuable feedback from our community, collaborators, and internal analysis, we are pleased to introduce the first iteration of Composable’s Change Management and Incident Response Process. This plan serves as an early blueprint for how we intend to manage incident procedures in the future. We view this as a dynamic process and are committed to continuous improvement. Your feedback is not only welcomed but actively encouraged as we refine this process.

Objective

To ensure continuous and informed communication with our community, we have instituted this Incident Management Process. This process is executed by our three specialized technical teams and aims to improve timelines and communications in managing any future planned maintenance upgrades or unplanned bugs and incidents.

Incident Identification

Key Performance Indicators (KPIs) will be established to monitor the state of our IBC connections between different ecosystems. These metrics will be carefully selected to provide real-time insights into system performance and health.

The system will be configured to trigger alerts based on these KPIs if they deviate from predefined thresholds. Alerts will be channeled to the appropriate teams via communication platforms such as Slack and PagerDuty.

Example KPIs for Relayer:

- Light client height
- Number of undelivered packets
- Number of undelivered acknowledgements

Example KPIs for Node:

- Node’s block height

Alerting Process

The following Composable teams will be notified through our alerting system:

- Engineering Teams (SRE, Bridging Team)
- Product and Operations
- Marketing and Community Management

Communication Strategy

Our engineering teams will promptly inform community managers about the status of any outages, should they occur. Community managers will then keep the Composable community updated through various channels such as Discord and Telegram, covering:

Unplanned service outages

- Current status of ongoing outages
- Estimated time for resolution

Planned maintenance activities

- Current status of maintenance
- Estimate time for maintenance timeline

Planned maintenance will be scheduled during weekdays (Monday to Thursday) and will be announced at least 48 hours in advance. These announcements will include details about when services are expected to be fully operational again.

We also implemented a new system on our discord that displays the status of our products.

- Operational
- Down
- Planned Maintenance

Incident Severity Levels

Critical: Transactions between ecosystems are not processing.

Warning: Delays in transaction processing.

Notifications

- In case of issues related to Picasso or Composable nodes (following teams will be notified: SRE, Parachain team)
- In case of issues related to Composable Cosmos Chain (Notional, SRE)
- In case of issues related to Relayers (SRE, Bridging team)

By adhering to these robust Incident Management Procedures, we aim to demonstrate our commitment to operational integrity and greatly improve our communication with our community.