informal
SYSTEMS

Security Audit Report

Injective Protocol: Protocol Design

Initial report: June 15, 2021

Issue revision: June 28, 2021

Contents

Audit overview

The Project

In May 2021, Injective engaged Informal Systems to conduct a security audit over the documentation and the current state of the implementation of *Injective Protocol*: a Cosmos-backed decentralized derivatives trading platform. The agreed-upon workplan consisted of two steps:

Milestone 1: Reviewing spot markets

The focus of this milestone was to review the code that implements exchange in the spot markets. As spot markets are relatively simple, we agreed that it was a good starting point. The input to this milestone was: documentation on Notion, code walkthrough, the codebase in the private github repository called `injective-core`. Deliverables include open issues that describe functional and security bugs as well as TLA+ specifications, which can be used for model-based testing.

In this milestone, we mainly focused on the audit of the `exchange` module.

Milestone 2: Reviewing derivative markets

The implementation of the derivative markets is more sophisticated in comparison to the spot markets. The input to this milestone was: documentation on Notion, the codebase in the private github repository called `injective-core`. Deliverables include open issues that describe functional and security bugs as well as TLA+ specifications, which can be used for model-based testing.

In this milestone, we mainly focused on the audit of the modules: `exchange`, `oracle`, and `insurance`.

Scope of this report

This report covers the audit in the framework of Milestones 1-2 that was conducted May 11 through June 14, 2021 by Informal Systems under the lead of Igor Konnov, with the support of Zarko Milosevic. The team spent 3 person-weeks on the audit.

As the codebase spans over 38 KLOC of Golang code, we could not perform an exhaustive audit of the whole codebase. Rather we have identified potential problems in the code and tried to trigger critical errors in the system.

Conducted work

Starting May 11, the Informal Systems team conducted an audit of the existing documentation and code in the project directory in the Cosmos repository of hash 4dac628e. The Injective Labs team was resolving issues that were blocking our further progress. Hence, we continued with more recent versions of the development branch.

The most important issues we documented in the findings which are part of this report, and as issues on the Injective Labs GitHub repository. A detailed list can be found in the Findings.

As we quickly found that the general code quality was high and the Injective Labs team tested their code on regular basis, we changed our auditing approach to model-based testing, which was backed by a symbolic model checker. To this end, we have designed high-level specifications of spot markets and derivative markets in TLA+, by following the English specifications that were provided by Injective Labs. Importantly, our TLA+ specifications do not focus on complete functional correctness. Rather, we used them to drive the system into a potentially problematic state that we could manually inspect, in order to trigger bugs in the system.

Findings

We have found that Injective Protocol is written with attention to details. Large parts of the codebase contain all necessary validation tests and do not let an attacker to easily exploit overflows, replay previously recorded transactions or perform timing attacks. As a result, our straightforward attempts to attack the system did not succeed.

As we switched to semi-automated model-based testing, we found issues with the command-line interface of the Injective Protocol, **all resolved**:

- IF-INJECTIVE-01,
- IF-INJECTIVE-04,
- IF-INJECTIVE-05,
- IF-INJECTIVE-06,
- IF-INJECTIVE-09.

None of these issues is severe, as they only affected the client interface. The main reason for the team paying less attention to CLI is that they are testing their system by running end-to-end integration tests (that do not use CLI) as well as manual testing via the web interface. The Injective Labs team was surprisingly responsive in fixing the discovered issues. Usually, they fixed issues in less than 1 hour after receiving a report on GitHub and the Discord channel. Hence, although CLI issues slightly impeded our progress, they did not block us.

By code inspection, we have found that Injective Protocol implements reach functionality in `abci.go:BeginBlocker` and `abci.go:EndBlocker`. While errors in Cosmos transactions are automatically recovered by the Cosmos framework, by rolling back an offending transaction, errors in `BeginBlock` and `EndBlock` are not automatically recovered. Every such an error results in halting the consensus engine, which effectively means that all validators would have to patch the code and to coordinate in restarting the blockchain. We have documented this potential issue in IF-INJECTIVE-12. The team has confirmed that this indeed a potential severe issue that requires careful redesign of the code. Later, we indeed found attack vectors IF-INJECTIVE-10 and IF-INJECTIVE-11 that exploited this issue. We believe that these are only two instances of the general issue IF-INJECTIVE-12. Hence, the issues IF-INJECTIVE-10, IF-INJECTIVE-11, and IF-INJECTIVE-12 are the most severe. We recommend designing good defense mechanisms against them. Both issues 10 and 11 highlight interesting sources of errors, to which the team should pay further attention:

- IF-INJECTIVE-10 was triggered after market expiration, which could potentially last for weeks or months in production. Interestingly, the user had only to launch a market and wait, without performing any trading activity. **Resolved**.
- IF-INJECTIVE-11 was triggered by corrupt input from a price feed. As price feeds are outside of the designer's control, we recommended the team to carefully validate and filter price feeds. **Resolved**.

Two further issues were less severe, but they could probably result in fraud or loss of tokens:

- In issue IF-INJECTIVE-07, changing the status of a spot market resulted in launching another market instance. **Resolved**.
- In issue IF-INJECTIVE-08, demolishing a spot market resulted in outstanding orders (and their tokens) being frozen. **Resolved**.

Finally, we found two non-critical issues:

- Transactions invoked by CLI contained a hard-coded recipient address: IF-INJECTIVE-02. **Resolved**.
- Transaction panic IF-INJECTIVE-03. **Resolved**.

We emphasize that the five severe issues would not be found by the standard lightweight static analysis or fuzzing. They required knowledge of the source code and executing carefully crafted sequences of transactions. We do not consider them as being easily exploitable.

Audit Dashboard

Target Summary

- **Name**: Injective Protocol
- **Version**: 4dac628eb1d08f4d66685e9f228f6ff53e9197c9 through baa69e1c366e9dc8727c7385fa120c08162b0
- **Type**: Implementation and preliminary documentation
- **Platform**: Golang

Engagement Summary

- **Dates**: May 11 through June 14, 2021 (kick-off meeting May 7)
- **Method**: Whitebox, model-based testing, symbolic model checking
- **Employees Engaged**: 2
- **Time Spent**: 21 person days

Fundings Summary by Severity and Difficulty

Severity	Difficulty	#	Finding
High	Low	1	IF-INJECTIVE-10
High	High	2	IF-INJECTIVE-11, IF-INJECTIVE-12
Medium	Medium	2	IF-INJECTIVE-07, IF-INJECTIVE-08
Low	Low	7	IF-INJECTIVE-02, IF-INJECTIVE-03, IF-INJECTIVE-01, IF-INJECTIVE-04, IF-INJECTIVE-05, IF-INJECTIVE-06, IF-INJECTIVE-09
Total		**12**	

Category Breakdown

Finding Type	#
Distributed System Reliability and Fault Tolerance	3
Protocol, Economics & Implementation	3
Implementation & Testing	6

Finding Type	#
Total	**12**

Severity Categories

Severity	Description
Informational	The issue does not pose an immediate risk (it is subjective in nature); they are typically suggestions around best practices or readability
Low	The issue is objective in nature, but the security risk is relatively small or does not represent security vulnerability
Medium	The issue is a security vulnerability that may not be directly exploitable or may require certain complex conditions in order to be exploited
High	The issue is exploitable security vulnerability

Difficulty Categories

Difficulty	Description
Low	Can be attacked by a user without special permission
Medium	Can be exploited without special permission with in-depth knowledge and control of the security architecture
High	Needs a collection of privileged users with in-depth knowledge and control of the security architecture

Engagement Goals

This audit was scoped by the Informal Systems team in order to assess the correctness and security of the Injective Protocol. It was planned along two Milestones. In the first milestone, the focus was to get familiarized with the codebase and look for potential attack vectors in the spot markets. In the second milestone, the focus was on derivative markets. As the scope of the project is too large for a short-term audit, we agreed that it was not feasible to acquire comprehensive understanding of the protocols and system functionality. Instead, we focused on potential attack scenarios by inspecting the code and running model-based tests.

Coverage

Informal Systems manually reviewed the documentation and code of the software in the Injective Chain directory starting at commit hash 4dac628eb1d08f4d66685e9f228f6ff53e9197c9. As the code was updated during the review, we continued with further commits through baa69e1c366e9dc8727c7385fa120c08162b08e0.

We focused on the backend Cosmos code in the modules: `exchange`, `insurance`, and `oracle`. As the codebase spans over 38 KLOC of Golang code, we could not perform an exhaustive audit of the whole codebase.

Recommendations

This section aggregates all the recommendations made during the audit. Short-term recommendations address the immediate causes of issues. Long-term recommendations pertain to the development process and long-term design goals.

Short term

- **Test for non-standard scenarios.** Issues IF-INJECTIVE-07 and IF-INJECTIVE-08 were probably outside of the standard scenarios of a testing engineer, as they are rarely used features.
- **Test for time-related issues.** As exemplified by IF-INJECTIVE-10, functional tests are not sufficient. Injective Protocol is using timeouts and hence it should be tested with timeouts in mind.
- **Avoid hard-coded addresses.** As exemplified by IF-INJECTIVE-02.
- **Add tests for CLI.** As demonstrated by issues IF-INJECTIVE-01, IF-INJECTIVE-04, IF-INJECTIVE-05, IF-INJECTIVE-06, IF-INJECTIVE-09.

Long term

- **ABCI methods.** As stressed in IF-INJECTIVE-12 and exemplified by IF-INJECTIVE-10, the system should be re-designed in a way that does not halt the consensus engine, if a panic occurs in the code that is triggered by the methods `abci.go:BeginBlock` and `abci.go:EndBlock`.
- **Price oracles.** As exemplified by IF-INJECTIVE-11, the team should pay attention to the interaction with the price oracles, as they are outside of the designer's control. Thus, price oracles may be used by an attacker or they can accidentally feed corrupt data into the system. We recommend receiving price values from `3f + 1` feeds and filtering out the `f` smallest and the `f` largest values, while averaging the rest.

Findings

#	Title	Type	Severity	Issue
IF-INJECTIVE-10	A sequence of transactions leads to a complete halt of consensus	Distributed System Reliability and Fault Tolerance	High	323
IF-INJECTIVE-11	Price feed does not validate prices, may crash consensus	Distributed System Reliability and Fault Tolerance	High	331
IF-INJECTIVE-12	Recommendation for recovery in EndBlocker	Distributed System Reliability and Fault Tolerance	High	301
IF-INJECTIVE-07	Changing the status of a spot market to Demolished introduces a market copy	Protocol, Economics & Implementation	Medium	302
IF-INJECTIVE-08	When a spot market is demolished the outstanding sell orders (and their coins) are frozen	Protocol, Economics & Implementation	Medium	304
IF-INJECTIVE-01	CLI interface fails with a stack trace when supplying incorrect arguments	Implementation & Testing	Low	287
IF-INJECTIVE-02	Hard-coded fee recipient in the client code	Protocol, Economics & Implementation	Low	289

#	Title	Type	Severity	Issue
IF-INJECTIVE-03	Missing validation tests in MsgInstantSpot-MarketLaunch, results in division by zero	Implementation & Testing	Low	291
IF-INJECTIVE-04	NPE when launching an instant spot market	Implementation & Testing	Low	295
IF-INJECTIVE-05	Outdated parameters in scripts/propose_spot_market.sh	Implementation & Testing	Low	296
IF-INJECTIVE-06	Incorrect parsing of arguments for injectived tx exchange create-spot-market-order	Implementation & Testing	Low	299
IF-INJECTIVE-09	CLI for launching derivative markets	Implementation & Testing	Low	322

IF-INJECTIVE-01

CLI interface fails with a stack trace when supplying incorrect arguments #287

Status: Resolved

Severity: Low

Type: Implementation & Testing

Difficulty: Low

Surfaced from Informal Systems audit at hash 4dac628eb1d08f4d66685e9f228f6ff53e9197c9.

Observed behavior

Here is an example for "query exchange deposits":

```
~/go/bin/injectived query exchange deposits
panic: runtime error: index out of range [1] with length 0
...
```

Expected behavior

An error message printed on stderr, without a stack trace. For instance, here is how the bank module reacts on the wrong number of arguments:

```
~/go/bin/injectived query bank balances
Error: accepts 1 arg(s), received 0
Usage:
  injectived query bank balances [address] [flags]
...
```

Version

Running `injectived` that was compiled from 4dac628eb1d08f4d66685e9f228f6ff53e9197c9.

IF-INJECTIVE-02

Hard-coded fee recipient in the client code #289

Status: Resolved

Severity: Low

Type: Protocol, Economics & Implementation

Difficulty: Low

Surfaced from Informal Systems audit of hash 4dac628eb1d08f4d66685e9f228f6ff53e9197c9

This is issue has been created for documentation purposes. It has been fixed by the team in 043a2402e59a985400c676dc6d4b6fa1ca85567b after communication on discord.

The client code contained a hard-coded address of the fee recipient: https://github.com/InjectiveLabs/injective-core/blob/4dac628eb1d08f4d66685e9f228f6ff53e9197c9/injective-chain/modules/exchange/client/cli/tx.go#L17 L188

The fix https://github.com/InjectiveLabs/injective-core/commit/043a2402e59a985400c676dc6d4b6fa1ca85567b sets the sender as the fee recipient.

IF-INJECTIVE-03

Missing validation tests in MsgInstantSpotMarketLaunch, results in division by zero #291

Status: Resolved

Severity: Low

Type: Implementation & Testing

Difficulty: Low

Surfaced from Informal Systems audit of hash 043a2402e59a985400c676dc6d4b6fa1ca85567b

It is possible to launch a market with incorrect parameters that results in a transaction panic later. Consider the following sequence of commands in the standard setup, as done with `./setup.sh`:

```
injectived tx exchange instant-spot-market-launch INJ/INJ inj inj \
  --from=genesis --chain-id=888 --keyring-backend=file
injectived tx exchange deposit 10000000inj --chain-id 888 \
  --from inj1cml96vmptgw99syqrrz8az79xer2pcgp0a885r
injectived tx exchange create-spot-limit-order buy INJ/INJ 10 10 \
  --from=user1 --chain-id=888 --keyring-backend=file
```

This leads to a transaction panic:

```
{"height":"2318",[...] to execute message; message index: 0: division by zero:
panic","logs":[],"info":"","gas_wanted":"200000","gas_used":"64317",
"tx":null,"timestamp":""}
```

The reason is that the market is launched with `min_price_tick_size = 0`:

```
injectived query exchange spot-markets
markets:
- base_denom: inj
  maker_fee_rate: "0.001000000000000000"
  market_id: 0x3b78a9b8efc920e7021cc30cb3c821df189585cc3eaa35d73ec8853a1780961d
  min_price_tick_size: "0.000000000000000000"
  min_quantity_tick_size: "0.000000000000000000"
  quote_denom: inj
  relayer_fee_share_rate: "1.000000000000000000"
  status: Active
  taker_fee_rate: "0.002000000000000000"
  ticker: INJ/INJ
```

IF-INJECTIVE-04

NPE when launching an instant spot market #295

Status: Resolved

Severity: Low

Type: Implementation & Testing

Difficulty: Low

Surfaced from Informal Systems audit of hash 1e4d2914b3ae616b98b05fb70eb487550fc99ed7

This is a follow up of #291. The recent fix in ba9f2eb7a76dfd81a9d1f74970597085e2253357 introduced NPE in the client. run the following command in the standard setup, as done with ./setup.sh:

```
injectived tx exchange instant-spot-market-launch INJ/INJ inj inj \
  --from=genesis --chain-id=888 --keyring-backend=file
Enter keyring passphrase:
panic: runtime error: invalid memory address or nil pointer dereference
[signal SIGSEGV: segmentation violation code=0x1 addr=0x0 pc=0x5fb461]

  ...
```

As far as I can tell, the code in `injective-chain/modules/exchange/client/cli/tx.go` fails to add the message fields `MinPriceTickSize` and `MinQuantityTickSize`, which results in an NPE later.

IF-INJECTIVE-05

Outdated parameters in scripts/propose_spot_market.sh #296

Status: Resolved

Severity: Low

Type: Implementation & Testing

Difficulty: Low

Surfaced from Informal Systems audit of hash 1e4d291

The script https://github.com/InjectiveLabs/injective-core/blob/dev/scripts/propose_spot_market.sh fails to launch a spot market:

```
./scripts/propose_spot_market.sh
Error: accepts 3 arg(s), received 5
Usage:
  injectived tx exchange spot-market-launch [ticker] [base_denom]
  [quote_denom] [flags]
...
```

I believe that the following line:

https://github.com/InjectiveLabs/injective-core/blob/1e4d2914b3ae616b98b05fb70eb487550fc99ed7/scripts/prop

should be replaced with:

```
yes $PASSPHRASE | injectived tx exchange spot-market-launch "$Ticker" \
"$BaseDenom" "$QuoteDenom"  --min-price-tick-size="$MaxPriceScaleDecimals" \
--min-quantity-tick-size="$MaxQuantityScaleDecimals" --title="$Title" \
--description="$Description" --deposit="100000000000inj" --from=genesis \
--chain-id=888 --keyring-backend=file --yes
```

IF-INJECTIVE-06

Incorrect parsing of arguments for injectived tx exchange create-spot-market-order #299

Status: Resolved

Severity: Low

Type: Implementation & Testing

Difficulty: Low

Surfaced from Informal Systems audit of hash 0189db186636991fd9076ee741d67ff05ae4c2c1

This is just a bug in functionality. The code below is using `args[1]` and `args[2]` for the quantity and price, whereas it should use `args[2]` and `args[3]`.

https://github.com/InjectiveLabs/injective-core/blob/0189db186636991fd9076ee741d67ff05ae4c2c1/injective-chain/modules/exchange/client/cli/tx.go#L263-L279

IF-INJECTIVE-07

Changing the status of a spot market to Demolished introduces a market copy #302

Status: Resolved

Severity: Medium

Type: Protocol, Economics & Implementation

Difficulty: Medium

Surfaced from Informal Systems IBC Audit of hash e39a091

Changing the market status to Demolished introduces a market copy

```
injectived tx exchange instant-spot-market-launch atom/inj atom inj \
  --min-price-tick-size=0.000000000000000001 \
  --min-quantity-tick-size=0.000000000000000001 --from=genesis \
  --chain-id=888 --keyring-backend=file --yes
injectived tx exchange deposit 10000.000000000000000000atom \
  --from=inj1cml96vmptgw99syqrrz8az79xer2pcgp0a885r --chain-id=888 \
  --keyring-backend=file --yes
injectived tx exchange create-spot-limit-order sell atom/inj \
  10000.000000000000000000 0.000000000000000001 --from=user1 --chain-id=888 \
  --keyring-backend=file --yes
injectived tx exchange set-spot-market-status \
  0xfbd55f13641acbb6e69d7b59eb335dabe2ecbfea136082ce2eedaba8a0c917a3 \
  Demolished  --title="atom/inj spot market status set" --description="XX" \
  --deposit="10000000000000000000000inj" --from=genesis --chain-id=888 \
  --keyring-backend=file --yes
injectived tx gov vote 1 yes --from=genesis --chain-id=888 \
  --keyring-backend=file --yes

injectived query exchange spot-markets
markets:
- base_denom: atom
  maker_fee_rate: "0.001000000000000000"
  market_id: 0xfbd55f13641acbb6e69d7b59eb335dabe2ecbfea136082ce2eedaba8a0c917a3
  min_price_tick_size: "0.000000000000000001"
  min_quantity_tick_size: "0.000000000000000001"
  quote_denom: inj
  relayer_fee_share_rate: "1.000000000000000000"
  status: Demolished
  taker_fee_rate: "0.002000000000000000"
  ticker: atom/inj
- base_denom: atom
  maker_fee_rate: "0.001000000000000000"
  market_id: 0xfbd55f13641acbb6e69d7b59eb335dabe2ecbfea136082ce2eedaba8a0c917a3
  min_price_tick_size: "0.000000000000000001"
  min_quantity_tick_size: "0.000000000000000001"
  quote_denom: inj
  relayer_fee_share_rate: "1.000000000000000000"
  status: Active
  taker_fee_rate: "0.002000000000000000"
  ticker: atom/inj
```

IF-INJECTIVE-08

When a spot market is demolished the outstanding sell orders (and their coins) are frozen #304

Status: Resolved

Severity: Medium

Type: Protocol, Economics & Implementation

Difficulty: Medium

Surfaced from Informal Systems IBC Audit of hash 8b31eedeea8e6b8fea63d656505ada62c788f587

Execute the following commands to create a sell order and demolish the market:

```
injectived tx exchange instant-spot-market-launch atom/inj atom inj \
  --min-price-tick-size=0.000000000000000001 \
  --min-quantity-tick-size=0.000000000000000001 --from=genesis \
  --chain-id=888 --keyring-backend=file --yes
injectived tx exchange deposit 10000.000000000000000000atom \
  --from=inj1cml96vmptgw99syqrrz8az79xer2pcgp0a885r --chain-id=888 \
  --keyring-backend=file --yes
injectived tx exchange create-spot-limit-order sell atom/inj \
  10000.000000000000000000 0.000000000000000001 --from=user1 --chain-id=888 \
  --keyring-backend=file --yes
injectived tx exchange set-spot-market-status \
  0xfbd55f13641acbb6e69d7b59eb335dabe2ecbfea136082ce2eedaba8a0c917a3 \
  Demolished  --title="atom/inj spot market status set" --description="XX" \
  --deposit="100000000000000000000inj" --from=genesis --chain-id=888 \
  --keyring-backend=file --yes
injectived tx gov vote 1 yes --from=genesis --chain-id=888 \
  --keyring-backend=file --yes
```

Now the market is demolished but `user1` has their coins still frozen:

```
injectived query exchange deposits inj1cml96vmptgw99syqrrz8az79xer2pcgp0a885r 0
deposits:
  atom:
    available_balance: "0.000000000000000000"
    total_balance: "10000.000000000000000000"
```

Moreover, user1 is not able to cancel the order, in order to retrieve the coins:

```
injectived tx exchange cancel-spot-limit-order "atom/inj" \
0xc6fe5d33615a1c52c08018c47e8bc53646a0e101000000000000000000000000 \
--from=user1  --chain-id=888 --keyring-backend=file --yes
injectived tx exchange cancel-spot-limit-order "atom/inj" \
0xc6fe5d33615a1c52c08018c47e8bc53646a0e101000000000000000000000000 \
--from=user1  --chain-id=888 --keyring-backend=file --yes
Enter keyring passphrase:
{"height":"275",
"txhash":"008037D8E2D3223B05FB36B9EB6FCE3F96CAA36F4419C71A2C7BFFCDE1AC0AF5",
"codespace":"exchange","code":4,"data":"","raw_log":"failed
to execute message; message index: 0: active spot market doesn't exist
0xfbd55f13641acbb6e69d7b59eb335dabe2ecbfea136082ce2eedaba8a0c917a3: spot market
not
found","logs":[],"info":"","gas_wanted":"200000","gas_used":"63531",
"tx":null,"timestamp":""}
```

IF-INJECTIVE-09

CLI for launching derivative markets #322

Status: Resolved

Severity: Low

Type: Implementation & Testing

Difficulty: Low

Surfaced from @informalsystems audit at hash e678a9f8c13090b353c3c1d042a1b0932ac3da38

CLI support for instant future markets seems to be outdated:

```
injectived tx exchange instant-expiryfuturesmarket-launch inj/atom inj \
  inj atom pricefeed 1623070240 --fees 10inj --from=genesis --chain-id=888 \
  --broadcast-mode=block --yes
Enter keyring passphrase:
Error: exchange fee cannot be nil: <nil>
```

IF-INJECTIVE-10

A sequence of transactions leads to a complete halt of consensus #323

Status: Resolved

Severity: High

Type: Distributed System Reliability and Fault Tolerance

Difficulty: Low

Surfaced from @informalsystems audit at hash e678a9f

This is a concrete sequence of transactions that triggers panic in `BeginBlocker` and halts consensus. Related to the theoretical possibility of panic in `EndBlocker` that was discussed in #301.

Here is the sequence of shell commands that should be executed on a clean installation that is initialized with `./setup.sh`:

```
EXPIRY=$((`date +%s`+60))
injectived tx oracle grant-price-feeder-privilege-proposal inj atom \
  inj1cml96vmptgw99syqrrz8az79xer2pcgp0a885r --deposit=10000000inj \
  --title="price feeder inj/atom" --description="price feeder inj/atom" \
  --from=user1 --chain-id=888 --broadcast-mode=block --yes
sleep 2
injectived tx gov vote 1 yes --from=genesis --chain-id=888 \
  --broadcast-mode=block --yes
sleep 15
injectived tx oracle relay-price-feed-price inj atom 0.000000000000000001 \
  --from=user1 --chain-id=888 --broadcast-mode=block --yes
injectived tx insurance create-insurance-fund --ticker=inj/atom \
  --quote-denom=inj --oracle-base=inj --oracle-quote=atom \
  --oracle-type=PriceFeed --expiry=$EXPIRY --initial-deposit=10000000inj \
  --from=genesis --chain-id=888 --broadcast-mode=block --yes
sleep 1
injectived tx exchange expiryfuturesmarket-launch inj/atom inj inj atom 2 \
  pricefeed $EXPIRY --title="launch inj/atom" --description="launch inj/atom" \
  --from=user1 --deposit=10000000inj --chain-id=888 --broadcast-mode=block --yes
```

```
sleep 2
injectived tx gov vote 2 yes --from=genesis --chain-id=888 \
  --broadcast-mode=block --yes
sleep 40
```

Once the futures market has expired (after 60 seconds), the code in `BeginBlocker` starts the settlement and panics on division by zero:

https://github.com/InjectiveLabs/injective-core/blob/457705b7c0fb95d562a27b68306c9ad060c5b344/injective-chain/modules/exchange/keeper/futures_settlement.go#L154-L156

See the log output below...

Recommendation

While we obviously recommend fixing the division by zero that is triggered by this sequence of commands, this issue demonstrates a more general concern. The code that is executed by `BeginBlock` is very sensitive to `panic`, which corrupts the application state and stops the node. We recommend to carefully examine the code that either explicitly calls `panic` or may trigger it (e.g., via methods of `sdk.Dec`) and provide reasonable protection against it. It's also desirable to keep only the necessary logic in `BeginBlock` and `EndBlock`.

Example of the server output:

```
NFO[0053] notifying bugsnag: CONSENSUS FAILURE!!!      INFO[0053] bugsnag.Notify:
not notifying in local       ERRO[0053] CONSENSUS FAILURE!!! err="division by
zero" module=consensus stack="goroutine 112 ... ...app.(*InjectiveApp).BeginBlocker(...)
```

IF-INJECTIVE-11

Price feed does not validate prices, may crash consensus #331

Status: resolved (as of June 15, 2021)

Severity: High

Type: Distributed System Reliability and Fault Tolerance

Difficulty: High

Surfaced from @informalsystems audit at hash baa69e1c366e9dc8727c7385fa120c08162b08e0

The crash is resolved in PR: https://github.com/InjectiveLabs/injective-core/pull/333

The general recommendation still applies.

pricefeed_msg_server.go does not validate the messages, so it is possible to feed it with arbitrary large (or small) values, e.g., 2^(255-60) - 1 and -2^(255-60)+1. When the price feed reports -2^(255-60)+1 several times, consensus crashes. Note that such values are not necessary a sign of an attack, but can originate from faulty software.

How to reproduce: Here is a sequence of commands to reproduce the issue on a clean installation (initialized with ./setup.sh):

```
injectived tx oracle grant-price-feeder-privilege-proposal inj atom \
  inj1jcltmuhplrdcwp7stlr4hlhlhgd4htqhe4c0cs --deposit=10000000inj \
  --title="price feeder inj/atom" --description="price feeder inj/atom" \
  --from=user2 --chain-id=injective-888 --broadcast-mode=block --yes
sleep 2
injectived tx gov vote 1 yes --from=genesis --chain-id=injective-888 \
  --broadcast-mode=block --yes
sleep 15

injectived tx sign --from=inj1jcltmuhplrdcwp7stlr4hlhlhgd4htqhe4c0cs \
  --chain-id=injective-888 --output-document=signed1.json unsigned1.json
injectived tx broadcast --broadcast-mode=block signed1.json

injectived tx exchange deposit 1000.000000000000000000inj \
  --from=inj1dzqd00lfd4y4qy2pxa0dsdwzfnmsu27hgttswz --chain-id=injective-888 \
  --broadcast-mode=block --yes

injectived tx insurance create-insurance-fund --ticker=inj/atom \
  --quote-denom=inj --oracle-base=inj --oracle-quote=atom \
  --oracle-type=PriceFeed --expiry=1623325840 --initial-deposit=10000000inj \
  --from=genesis --chain-id=injective-888 --broadcast-mode=block --yes
sleep 1

injectived tx exchange instant-expiry-futures-market-launch \
  --ticker=inj/atom --quote-denom=inj --oracle-base=inj --oracle-quote=atom \
  --oracle-type=PriceFeed --expiry=1623325840 --maker-fee-rate=0.001 \
  --taker-fee-rate=0.001 --initial-margin-ratio=0.05 \
  --maintenance-margin-ratio=0.02 --min-price-tick-size=0.000000000000000001 \
  --min-quantity-tick-size=0.000000000000000001 --from=user1 \
  --chain-id=injective-888 --broadcast-mode=block --yes
sleep 2

injectived tx exchange deposit 1.000000000000000000inj \
  --from=inj1jcltmuhplrdcwp7stlr4hlhlhgd4htqhe4c0cs --chain-id=injective-888 \
  --broadcast-mode=block --yes

injectived tx sign --from=inj1dzqd00lfd4y4qy2pxa0dsdwzfnmsu27hgttswz \
  --chain-id=injective-888 --output-document=signed2.json unsigned2.json
injectived tx broadcast --broadcast-mode=block signed2.json

injectived tx sign --from=inj1jcltmuhplrdcwp7stlr4hlhlhgd4htqhe4c0cs \
  --chain-id=injective-888 --output-document=signed3.json unsigned3.json
injectived tx broadcast --broadcast-mode=block signed3.json

injectived tx sign --from=inj1jcltmuhplrdcwp7stlr4hlhlhgd4htqhe4c0cs \
  --chain-id=injective-888 --output-document=signed4.json unsigned4.json
injectived tx broadcast --broadcast-mode=block signed4.json

injectived tx sign --from=inj1jcltmuhplrdcwp7stlr4hlhlhgd4htqhe4c0cs \
  --chain-id=injective-888 --keyring-backend file unsigned10.json \
  >signed10.json && injectived tx broadcast signed10.json  -b block
```

(The above sequence is most likely not the shortest one possible.)

The unsigned json files are like follows:

```
cat unsigned1.json
{"body": {"messages": [{"@type":
  "/injective.oracle.v1beta1.MsgRelayPriceFeedPrice",
  "sender": "inj1jcltmuhplrdcwp7stlr4hlhlhgd4htqhe4c0cs",
  "base": ["inj"], "quote": ["atom"], "price": ["1.000000000000000000"]}],
  "memo": "", "timeout_height": "0", "extension_options": [],
  "non_critical_extension_options": []}, "auth_info": {"signer_infos": [],
  "fee": {"amount": [], "gas_limit": "200000", "payer": "", "granter": ""}},
  "signatures": []}

cat unsigned2.json

{"body": {"messages": [{"@type":
"/injective.exchange.v1beta1.MsgCreateDerivativeLimitOrder", "sender":
"inj1dzqd00lfd4y4qy2pxa0dsdwzfnmsu27hgttswz", "order": {"market_id":
"0x7b01f008f84e7b87c93dc69efc0a0d860f09a17c5024e10de7b024dca45066bb",
"order_info": {"subaccount_id":
"0x6880D7bfE96D49501141375ED835C24cf70E2bD7000000000000000000000000",
"fee_recipient": "inj1dzqd00lfd4y4qy2pxa0dsdwzfnmsu27hgttswz", "price":
"0.000000000000000001", "quantity": "0.000000000000000001"}, "order_type":
"SELL", "margin": "0.000000000000000001", "trigger_price": null}}], "memo": "",
"timeout_height": "0", "extension_options": [],
"non_critical_extension_options": []}, "auth_info": {"signer_infos": [], "fee":
{"amount": [], "gas_limit": "200000", "payer": "", "granter": ""}},
"signatures": []}

cat unsigned3.json

{"body": {"messages": [{"@type":
"/injective.exchange.v1beta1.MsgCreateDerivativeLimitOrder", "sender":
"inj1jcltmuhplrdcwp7stlr4hlhlhgd4htqhe4c0cs", "order": {"market_id":
"0x7b01f008f84e7b87c93dc69efc0a0d860f09a17c5024e10de7b024dca45066bb",
"order_info": {"subaccount_id":
"0x963EBDf2e1f8DB8707D05FC75bfeFFBa1B5BaC17000000000000000000000000",
"fee_recipient": "inj1jcltmuhplrdcwp7stlr4hlhlhgd4htqhe4c0cs", "price":
"0.000000000000000001", "quantity": "0.000000000000000001"}, "order_type":
```

```
"BUY", "margin": "0.000000000000000021", "trigger_price": null}}], "memo": "",
"timeout_height": "0", "extension_options": [],
"non_critical_extension_options": []}, "auth_info": {"signer_infos": [], "fee":
{"amount": [], "gas_limit": "200000", "payer": "", "granter": ""}},
"signatures": []}

cat unsigned4.json

{"body": {"messages": [{"@type":
"/injective.oracle.v1beta1.MsgRelayPriceFeedPrice", "sender":
"inj1jcltmuhplrdcwp7stlr4hlhlhgd4htqhe4c0cs", "base": ["inj"], "quote":
["atom"], "price":
["-289480223093290488558927462521719769633174961664101410098643.396001978282409983"]}],
"memo": "", "timeout_height": "0", "extension_options": [],
"non_critical_extension_options": []}, "auth_info": {"signer_infos": [], "fee":
{"amount": [], "gas_limit": "200000", "payer": "", "granter": ""}},
"signatures": []}

cat unsigned10.json

{"body": {"messages": [{"@type":
"/injective.oracle.v1beta1.MsgRelayPriceFeedPrice", "sender":
"inj1jcltmuhplrdcwp7stlr4hlhlhgd4htqhe4c0cs", "base": ["inj"], "quote":
["atom"], "price":
["-502168138830934461106863153856613313288188435557122761031677.999999999999999999"]}],
"memo": "", "timeout_height": "0", "extension_options": [],
"non_critical_extension_options": []}, "auth_info": {"signer_infos": [], "fee":
{"amount": [], "gas_limit": "200000", "payer": "", "granter": ""}},
"signatures": []}
```

Example of the output in the server log:

```
ERRO[0256] CONSENSUS FAILURE!!!                          err="decimal out of range;
got: 259, max: 255" module=consensus stack="goroutine 114 ... ...oracle/keeper.Keeper.Ge
0x0, 0x0, 0x0, 0x0, 0x0, 0x2c74170, 0xc001478a50, 0x2cb3678, 0xc000e0c8d0,
```

Recommendation: Validate and filter the input that is received from the price oracles. For instance, there is probably no economic sense in negative prices. If it is not clear how to constrain the prices, you could receive price values from `3 * f + 1` oracles, throw away the `f` smallest values and the `f` largest values and average the rest. By doing so you can deal with Byzantine oracles.

IF-INJECTIVE-12

Recommendation for recovery in EndBlocker #301

Status: Unresolved (as of June 11, 2021)

Severity: High

Type: Distributed System Reliability and Fault Tolerance

Difficulty: High

Surfaced from Informal Systems IBC Audit of hash e39a091197a1d8178edbce863a88e0452aa3443d

The function `EndBlocker` in `abci.go` runs the core logic of the exchange module:

https://github.com/InjectiveLabs/injective-core/blob/3c78e3962dbf3aa51fd3c283528ce6e5c6ce7602/injective-chain/modules/exchange/abci.go#L19-L139

The code that is called in `EndBlocker` can potentially call `panic`. For instance, the code in `sdk.Dec` may panic on overflow. In contrast to IBC handlers, the code in `EndBlocker` does not recover from panic. Instead, the Tendermint consensus stops operating. It is easy to see the potential effect of panic in `EndBlocker` by adding an explicit call to `panic` in `EndBlocker`.

Recommendation

While we did not manage to find a set of transactions that would trigger panic in `EndBlocker` (e.g., by triggering an overflow in `Dec`), we recommend wrapping the code in `EndBlocker` with a recovery block. Given the complexity of the logic in `EndBlocker`, it is not clear to us, whether such a recovery would be easy to implement.

```
───────────────────── MODULE spots ─────────────────────
EXTENDS Integers, Sequences, typedefs

CONSTANTS
     accounts in the system
     @type: Set(ACCOUNT);
    ACCOUNTS,
     available coin types
     @type: Set(COIN);
    COINS,
     potential markets
     @type: Set(⟨COIN, COIN⟩);
    MARKETS

 18 places are reserved for the digits after "."
PRECISION ≜ 10^18

VARIABLES
     cosmos account balances
     @type: ⟨ACCOUNT, COIN⟩ → Int;
    balances,
     available deposits on subaccounts
     @type: ⟨ACCOUNT, COIN⟩ → Int;
    available,
     total deposits on subaccounts
     @type: ⟨ACCOUNT, COIN⟩ → Int;
    total,
     available markets
     @type: Set(MARKET);
    running_markets,
     whether there was a failing transaction
     @type: Bool;
    tx_fail,
     the last executed tx
     @type: TX;
    tx

Init ≜
    ∧ tx = [type ↦ "init", fail ↦ FALSE]
     these are the balances in setup-informal.sh, which we can change later
    ∧ balances = [a ∈ ACCOUNTS, c ∈ COINS ↦
          IF a = "user1" ∨ c = "inj"
           THEN  1000000000000000000000.000000000000000000
                 1000000000000000000000000000000000000000
           ELSE  0
      ]
    $\land$ $available = [a \in ACCOUNTS,\ c \in COINS \mapsto 0]$
    $\land$ $total = [a \in ACCOUNTS,\ c \in COINS \mapsto 0]$
    $\land$ $running\_markets = \{\}$
    $\land$ $tx\_fail = \text{FALSE}$

$Deposit(a,\ c) \triangleq$
    $\exists\, quantity \in Int :$
        LET $fail \triangleq$ $\lor$ $balances[a,\ c] < quantity$
                      $\lor$ $quantity < PRECISION$
                      $\lor$ $quantity \% PRECISION \neq 0$
        IN
        $\land$ $balances' = [balances$ EXCEPT $![a,\ c] = @ - quantity]$
        $\land$ $available' = [available$ EXCEPT $![a,\ c] = @ + quantity]$
        $\land$ $total' = [total$ EXCEPT $![a,\ c] = @ + quantity]$
        $\land$ $tx' = [type \mapsto$ "deposit",
                  $fail \mapsto fail,$
                  $coin \mapsto c,$
                  $account \mapsto a,$
                  $quantity \mapsto quantity]$
        $\land$ $tx\_fail' = (fail \lor tx\_fail)$
        $\land$ UNCHANGED $running\_markets$

$Withdraw(a,\ c) \triangleq$
    $\exists\, quantity \in Int :$
        LET $fail \triangleq$ $\lor$ $available[a,\ c] < quantity$
                      $\lor$ $quantity < PRECISION$
                      $\lor$ $quantity \% PRECISION \neq 0$
        IN
        $\land$ $balances' = [balances$ EXCEPT $![a,\ c] = @ + quantity]$
        $\land$ $available' = [available$ EXCEPT $![a,\ c] = @ - quantity]$
        $\land$ $total' = [total$ EXCEPT $![a,\ c] = @ - quantity]$
        $\land$ $tx' = [type \mapsto$ "withdraw",
                  $fail \mapsto fail,$
                  $coin \mapsto c,$
                  $account \mapsto a,$
                  $quantity \mapsto quantity]$
        $\land$ $tx\_fail' = (fail \lor tx\_fail)$
        $\land$ UNCHANGED $running\_markets$

$LaunchSpotMarket(a,\ m) \triangleq$
    $\land$ $m \notin running\_markets$
    $\land$ $running\_markets' = \{m\} \cup running\_markets$
    $\land$ $tx' = [type \mapsto$ "instant-spot-market-launch",
              $fail \mapsto \text{FALSE},$
              $base \mapsto m[1],$
              $quote \mapsto m[2],$
```

```
            $account \mapsto a]$
    $\land$ UNCHANGED $\langle balances, available, total, tx\_fail \rangle$

$DemolishSpotMarket(a, m) \triangleq$
    $\land m \in running\_markets$
    $\land running\_markets' = running\_markets \setminus \{m\}$
    $\land tx' = [type \mapsto$ "set-spot-market-status",
            $fail \mapsto$ FALSE,
            $base \mapsto m[1]$,
            $quote \mapsto m[2]$,
            $status \mapsto$ "Demolished",
            $account \mapsto a]$
    $\land$ UNCHANGED $\langle balances, available, total, tx\_fail \rangle$

$PayPlusFee(coins) \triangleq$
     we make sure that the user has enough coins to pay the taker fee
    $(1002 * coins) \div 1000$

 @type: $(Str, \langle Str, Str \rangle) \Rightarrow Bool$;
$CreateSpotLimitOrderBuy(a, m) \triangleq$
    LET $base \triangleq m[1]$
        $quote \triangleq m[2]$
    IN
    $\exists quantity, price \in Int :$
      LET $quote\_quantity \triangleq quantity * price$
          $ppf \triangleq PayPlusFee(quote\_quantity)$
          $fail \triangleq \lor available[a, quote] < ppf$
                  $\lor quantity \leq 0$
                  $\lor price \leq 0$
      IN
      $\land m \in running\_markets$
      $\land available' = [available$ EXCEPT $![a, quote] = @ - ppf]$
      $\land tx' = [type \mapsto$ "create-spot-limit-order",
              $fail \mapsto fail$,
              $direction \mapsto$ "buy",
              $base \mapsto base$,
              $quote \mapsto quote$,
              $quantity \mapsto quantity$,
              $price \mapsto price$,
              $account \mapsto a$
             ]
      $\land tx\_fail' = (fail \lor tx\_fail)$
      $\land$ UNCHANGED $\langle balances, running\_markets, total \rangle$

 @type: $(Str, \langle Str, Str \rangle) \Rightarrow Bool$;
$CreateSpotLimitOrderSell(a, m) \triangleq$
```

```
    LET $base \triangleq m[1]$
        $quote \triangleq m[2]$
    IN
    $\exists\, quantity, price \in Int$ :
      LET $fail \triangleq available[a, base] < quantity \lor quantity \leq 0 \lor price \leq 0$ IN
      $\land\ m \in running\_markets$
      $\land\ available' = [available$ EXCEPT $![a, base] = @ - quantity]$
      $\land\ tx' = [type \mapsto$ "create-spot-limit-order",
                  $fail \mapsto fail$,
                  $direction \mapsto$ "sell",
                  $base \mapsto base$,
                  $quote \mapsto quote$,
                  $quantity \mapsto quantity$,
                  $price \mapsto price$,
                  $account \mapsto a$
                 ]
      $\land\ tx\_fail' = (fail \lor tx\_fail)$
      $\land$ UNCHANGED $\langle balances, running\_markets, total \rangle$

$Next \triangleq$
    $\lor\ \exists\, a \in ACCOUNTS \setminus \{$"genesis"$\}, c \in COINS$ :
        $\lor\ Deposit(a, c)$
        $\lor\ Withdraw(a, c)$
    $\lor\ \exists\, m \in MARKETS$ :
        $\lor\ LaunchSpotMarket($"genesis"$, m)$
        $\lor\ DemolishSpotMarket($"genesis"$, m)$
    $\lor\ \exists\, a \in ACCOUNTS \setminus \{$"genesis"$\}, m \in MARKETS$ :
        $\lor\ CreateSpotLimitOrderBuy(a, m)$
        $\lor\ CreateSpotLimitOrderSell(a, m)$

  restrict to non-failing actions only
$NextNoFail \triangleq$
    $Next \land \neg tx\_fail'$
```

```
──────────────── MODULE MC_spots ────────────────
EXTENDS FiniteSets, typedefs

ACCOUNTS ≜ {"user1", "user2", "genesis"}
COINS ≜ {"inj", "atom"}

 @type: (Str, Str) ⇒ ⟨Str, Str⟩;
pair(i, j) ≜ ⟨i, j⟩

MARKETS ≜ {pair("inj", "atom"), pair("atom", "inj")}

VARIABLES
      cosmos account balances
      @type: ⟨ACCOUNT, COIN⟩ → Int;
    balances,
      available deposits on subaccounts
      @type: ⟨ACCOUNT, COIN⟩ → Int;
    available,
      total deposits on subaccounts
      @type: ⟨ACCOUNT, COIN⟩ → Int;
    total,
      available markets
      @type: Set(MARKET);
    running_markets,
      whether there was a failing transaction
      @type: Bool;
    tx_fail,
      the last executed tx
      @type: TX;
    tx

INSTANCE spots

NoSpotLimit ≜
    tx.type ≠ "create-spot-limit-order"

NoAtoms ≜
    tx_fail ∨ available["user1", "atom"] = 0

SomeAtoms ≜
    tx_fail ∨ available["user1", "atom"] ≥ total["user1", "atom"]

 @type: Seq(STATE ) ⇒ Bool;
TraceInvManyOrders(hist) ≜
    LET Example ≜
          LET  @type: (TX, ACCOUNT) ⇒ Bool;
              IsBuy(ptx, acc) ≜
              ∧ ptx.type = "create-spot-limit-order"
```

```
            $\land\ ptx.direction = $ “buy”
            $\land\ ptx.account = acc$
        IN
        LET   @type: $(TX,\ \text{ACCOUNT}) \Rightarrow Bool$;
            $IsSell(ptx,\ acc)\ \triangleq$
            $\land\ ptx.type = $ “create-spot-limit-order”
            $\land\ ptx.direction = $ “sell”
            $\land\ ptx.account = acc$
        IN
        $\land\ Cardinality(\{i \in \text{DOMAIN}\ hist : IsBuy(hist[i].tx,$ “user1”$)\}) \geq 1$
        $\land\ Cardinality(\{i \in \text{DOMAIN}\ hist : IsSell(hist[i].tx,$ “user1”$)\}) \geq 1$
        $\land\ Cardinality(\{i \in \text{DOMAIN}\ hist : IsBuy(hist[i].tx,$ “user2”$)\}) \geq 1$
    IN
    $\neg Example$

 @type: $Seq(\text{STATE}) \Rightarrow Bool$;
$TraceInv1(hist)\ \triangleq$
    $\lor\ Len(hist) < 4$
    $\lor$  LET $Example\ \triangleq$
            $\land\ \forall\, i \in \text{DOMAIN}\ hist :$
                $\neg hist[i].tx.fail$
            $\land\ \exists\, i \in \text{DOMAIN}\ hist :$
                $hist[i].tx.type = $ “deposit” $\land\ hist[i].tx.coin = $ “inj”
            $\land\ \exists\, i \in \text{DOMAIN}\ hist :$
                $hist[i].tx.type = $ “withdraw” $\land\ hist[i].tx.coin = $ “inj”
        IN
        $\neg Example$

 @type: $Seq(\text{STATE}) \Rightarrow Bool$;
$TraceInvBuySell(hist)\ \triangleq$
    $\lor\ hist[Len(hist)].tx\_fail$
    $\lor\ \neg\exists\, i,\ j \in \text{DOMAIN}\ hist :$
        $\land\ hist[i].tx.type = $ “create-spot-limit-order”
        $\land\ hist[j].tx.type = $ “create-spot-limit-order”
        $\land\ hist[i].tx.direction = $ “buy”
        $\land\ hist[j].tx.direction = $ “sell”
        $\land\ hist[i].tx.account \neq hist[j].tx.account$
        $\land\ hist[i].tx.base = hist[j].tx.base$
        $\land\ hist[i].tx.quote = hist[j].tx.quote$
        $\land\ hist[i].tx.quantity > hist[j].tx.quantity + PRECISION$
        $\land\ hist[i].tx.price \geq hist[j].tx.price + PRECISION$

 @type: $Seq(\text{STATE}) \Rightarrow Bool$;
$TraceInvOutstandingSell(hist)\ \triangleq$
    LET $Violation\ \triangleq$
      $\land\ \neg hist[Len(hist)].tx\_fail$
    $\land$ LET $last \triangleq hist[Len(hist)]$ IN
        $\land$ $last.tx.type =$ "create-spot-limit-order"
        $\land$ $last.tx.direction =$ "sell"
        $\land$ $last.tx.quantity \geq 10000 * PRECISION$
    $\land$ $\forall i \in$ DOMAIN $hist$ :
        $\lor$ $hist[i].tx.type \neq$ "create-spot-limit-order"
        $\lor$ $hist[i].tx.direction \neq$ "buy"
  IN
  $\lnot Violation$
```

@type: Seq(STATE) $\Rightarrow$ $Bool$;

```
$TraceInvOutstandingSellAndDemolished(hist) \triangleq$
  LET $Violation \triangleq$
    $\land \lnot hist[Len(hist)].tx\_fail$
    $\land Len(hist) > 2$
    $\land$ LET $sell \triangleq hist[Len(hist) - 1]$ IN
        $\land$ $sell.tx.type =$ "create-spot-limit-order"
        $\land$ $sell.tx.direction =$ "sell"
        $\land$ $sell.tx.quantity \geq 10000 * PRECISION$
    $\land$ LET $demolish \triangleq hist[Len(hist)]$ IN
        $\land$ $demolish.tx.type =$ "set-spot-market-status"
        $\land$ $demolish.tx.status =$ "Demolished"
    $\land$ $\forall i \in$ DOMAIN $hist$ :
        $\lor$ $hist[i].tx.type \neq$ "create-spot-limit-order"
        $\lor$ $hist[i].tx.direction \neq$ "buy"
  IN
  $\lnot Violation$
```

use this view to enumerate various scenarios

```
$TxView \triangleq$
  $\langle tx.type, tx\_fail \rangle$
```

MODULE *typedefs*

@*typeAlias*: $ACCOUNT = Str$;
@*typeAlias*: $COIN = Str$;
@*typeAlias*: $BALANCE = \langle ACCOUNT,\ COIN \rangle \to Int$;
@*typeAlias*: $MARKET = \langle COIN,\ COIN \rangle$;
@*typeAlias*: $TX =$ [type: Str, fail: $Bool$, account: ACCOUNT, coin: COIN,
base: COIN, quote: COIN, quantity: Int, price: Int,
direction: Str, status: Str];
@*typeAlias*: STATE = [balances: BALANCE, available: BALANCE, total: BALANCE,
tx: TX, $running\_markets$: $Set(MARKET)$, tx_fail: $Bool$];

a dummy definition to define aliases
$typedefs\_aliases \triangleq$ FALSE

```
──────────────────────── MODULE futures ────────────────────────
EXTENDS Integers, Sequences, typedefs_futures

CONSTANTS
     accounts in the system
     @type: Set(ACCOUNT);
    ACCOUNTS,
     available coin types
     @type: Set(COIN);
    COINS,
     potential markets
     @type: Set(⟨COIN, COIN⟩);
    MARKETS

 18 places are reserved for the digits after “.”
PRECISION ≜ 10^18

 the deposit that has to be put on a proposal
DEPOSIT ≜ 10000000

 the initial margin ratio when an order is placed, sync with atomkraft.py
INITIAL_MARGIN_RATIO ≜ (5 ∗ PRECISION) ÷ 100

 the initial margin ratio when an order is placed, sync with atomkraft.py
MAINTENANCE_MARGIN_RATIO ≜ (2 ∗ PRECISION) ÷ 100

 the fee in ’inj’ for launching an instant market
LISTING_FEE ≜ 1000000000000000000000

VARIABLES
     cosmos account balances
     @type: ⟨ACCOUNT, COIN⟩ → Int;
    balances,
     available deposits on subaccounts
     @type: ⟨ACCOUNT, COIN⟩ → Int;
    available,
     total deposits on subaccounts
     @type: ⟨ACCOUNT, COIN⟩ → Int;
    total,
     available markets
     @type: Set(MARKET);
    running_markets,
     status of a price feed
     @type: MARKET → Str;
    active_feeds,
     market prices as reported by the price feed
     @type: MARKET → Int;
    $prices$,
     whether there was a failing transaction
     @type: $Bool$;
    $tx\_fail$,
     the last executed $tx$
     @type: $TX$;
    $tx$

$Init \triangleq$
    $\land\ tx = [type \mapsto$ "init", $fail \mapsto$ FALSE$]$
    these are the balances in setup-$informal.sh$, which we can change later
    $\land\ balances = [a \in ACCOUNTS,\ c \in COINS \mapsto$
         IF $a =$ "user1" $\land\ c =$ "atom"
          THEN $1000000000000000000000000$
          ELSE IF $c =$ "inj"
                THEN  1000.000000000000000000
                      $1000000000000000000000$
                ELSE  $0$
      $]$
    $\land\ available = [a \in ACCOUNTS,\ c \in COINS \mapsto 0]$
    $\land\ total = [a \in ACCOUNTS,\ c \in COINS \mapsto 0]$
    $\land\ running\_markets = \{\}$
    $\land\ active\_feeds = [m \in MARKETS \mapsto$ ""$]$
    $\land\ prices = [m \in MARKETS \mapsto 0]$
    $\land\ tx\_fail =$ FALSE

$Deposit(a,\ c) \triangleq$
    $\exists\ quantity \in Int :$
      LET $fail \triangleq \lor\ balances[a,\ c] < quantity$
                 $\lor\ quantity < PRECISION$
                 $\lor\ quantity \% PRECISION \neq 0$
      IN
       $\land\ balances' = [balances$ EXCEPT $![a,\ c] = @ - quantity]$
       $\land\ available' = [available$ EXCEPT $![a,\ c] = @ + quantity]$
       $\land\ total' = [total$ EXCEPT $![a,\ c] = @ + quantity]$
       $\land\ tx' = [type \mapsto$ "deposit",
               $fail \mapsto fail$,
               $coin \mapsto c$,
               $account \mapsto a$,
               $quantity \mapsto quantity]$
       $\land\ tx\_fail' = (fail \lor tx\_fail)$
       $\land$ UNCHANGED $\langle running\_markets,\ active\_feeds,\ prices \rangle$

$Withdraw(a,\ c) \triangleq$
    $\exists\ quantity \in Int :$
      LET $fail \triangleq \lor\ available[a,\ c] < quantity$
```

```
            $\lor$ $quantity < PRECISION$
            $\lor$ $quantity \% PRECISION \neq 0$
    IN
    $\land$ $balances' = [balances$ EXCEPT $![a, c] = @ + quantity]$
    $\land$ $available' = [available$ EXCEPT $![a, c] = @ - quantity]$
    $\land$ $total' = [total$ EXCEPT $![a, c] = @ - quantity]$
    $\land$ $tx' = [type \mapsto$ "withdraw",
                $fail \mapsto fail$,
                $coin \mapsto c$,
                $account \mapsto a$,
                $quantity \mapsto quantity]$
    $\land$ $tx\_fail' = (fail \lor tx\_fail)$
    $\land$ UNCHANGED $\langle running\_markets, active\_feeds, prices \rangle$

$LaunchFuturesMarket(a, m) \triangleq$
  LET $fail \triangleq$
        $\lor$ $m \in running\_markets$
        $\lor$ $active\_feeds[m] =$ ""
        $\lor$ $balances[a, m[1]] < DEPOSIT$
  IN
  $\land$ $running\_markets' = \{m\} \cup running\_markets$
  $\land$ $tx' = [type \mapsto$ "expiryfuturesmarket-launch",
              $fail \mapsto fail$,
              $base \mapsto m[1]$,
              $quote \mapsto m[2]$,
              $account \mapsto a]$
  $\land$ $balances' = [balances$ EXCEPT $![a, m[1]] = @ - DEPOSIT]$
  $\land$ $tx\_fail' = (fail \lor tx\_fail)$
  $\land$ UNCHANGED $\langle available, total, active\_feeds, prices \rangle$

$LaunchInstantFuturesMarket(a, m) \triangleq$
  LET $fail \triangleq$
        $\lor$ $m \in running\_markets$
        $\lor$ $active\_feeds[m] =$ ""
        $\lor$ $prices[m] = 0$
        $\lor$ $balances[a,$ "inj"$] < LISTING\_FEE$
  IN
  $\land$ $running\_markets' = \{m\} \cup running\_markets$
  $\land$ $tx' = [type \mapsto$ "instant-expiry-futures-market-launch",
              $fail \mapsto fail$,
              $base \mapsto m[1]$,
              $quote \mapsto m[2]$,
              $account \mapsto a]$
  $\land$ $balances' = [balances$ EXCEPT $![a,$ "inj"$] = @ - LISTING\_FEE]$
  $\land$ $tx\_fail' = (fail \lor tx\_fail)$
```

```
    $\land$ UNCHANGED $\langle available, total, active\_feeds, prices \rangle$

$GrantPriceFeeder(a, m) \triangleq$
    LET $fail \triangleq active\_feeds[m] \neq$ "" $\lor balances[a,$ "inj"$] < DEPOSIT$ IN
    $\land tx' = [type \mapsto$ "grant-price-feeder-privilege-proposal",
              $fail \mapsto fail,$
              $base \mapsto m[1],$
              $quote \mapsto m[2],$
              $account \mapsto a]$
    $\land tx\_fail' = (fail \lor tx\_fail)$
    $\land balances' = [balances$ EXCEPT $![a,$ "inj"$] = @ - DEPOSIT]$
    $\land active\_feeds' = [active\_feeds$ EXCEPT $![m] = a]$
    $\land$ UNCHANGED $\langle available, total, running\_markets, prices \rangle$

@type: (ACCOUNT, MARKET) $\Rightarrow Bool$;
$RelayPrice(a, m) \triangleq$
    $\exists price \in Int :$
      LET $fail \triangleq active\_feeds[m] \neq a$ IN
      $\land tx' = [type \mapsto$ "relay-price-feed",
            $fail \mapsto fail,$
            $base \mapsto m[1],$
            $quote \mapsto m[2],$
            $price \mapsto price,$
            $account \mapsto a]$
      $\land tx\_fail' = (fail \lor tx\_fail)$
      $\land prices' = [prices$ EXCEPT $![m] = price]$
      $\land$ UNCHANGED $\langle balances, available, total, running\_markets, active\_feeds \rangle$

$PayPlusFee(coins) \triangleq$
      we make sure that the user has enough coins to pay the taker fee
    $(1002 * coins) \div 1000$

this is a magic formula from the notion spec
$IsBuyMargin(margin, price, quantity, market) \triangleq$
    LET $markPrice \triangleq prices[market]$ IN
    $\land margin * PRECISION \geq (quantity * INITIAL\_MARGIN\_RATIO * price)$
    $\land margin * PRECISION \geq$
      $quantity * ((INITIAL\_MARGIN\_RATIO * markPrice)$
            $- (markPrice + price) * PRECISION)$

this is a magic formula from the notion spec
$IsSellMargin(margin, price, quantity, market) \triangleq$
    LET $markPrice \triangleq prices[market]$ IN
    $\land margin * PRECISION \geq quantity * INITIAL\_MARGIN\_RATIO * price$
    $\land margin * PRECISION \geq$
      $quantity * (INITIAL\_MARGIN\_RATIO * markPrice)$
```

```
        $- (price + markPrice) * PRECISION$

@type: $(Str, \langle Str, Str \rangle) \Rightarrow Bool$;
$CreateDerivativeLimitOrderBuy(a, m) \triangleq$
  LET $base \triangleq m[1]$
      $quote \triangleq m[2]$
  IN
  $\exists\, quantity, price, margin \in Int :$
    LET $fail \triangleq \lor\, available[a, base] \leq margin$
                 $\lor\, quantity \leq 0$
                 $\lor\, price \leq 0$
                 $\lor\, \neg IsBuyMargin(margin, price, quantity, m)$
    IN
    $\land\, m \in running\_markets$
    $\land\, available' = [available$ EXCEPT $![a, quote] = @ - margin]$
    $\land\, tx' = [type \mapsto$ "create-derivative-limit-order",
            $fail \mapsto fail$,
            $direction \mapsto$ "buy",
            $base \mapsto base$,
            $quote \mapsto quote$,
            $quantity \mapsto quantity$,
            $price \mapsto price$,
            $margin \mapsto margin$,
            $account \mapsto a$
           ]
    $\land\, tx\_fail' = (fail \lor tx\_fail)$
    $\land$ UNCHANGED $\langle balances, running\_markets, total, active\_feeds, prices \rangle$

@type: $(Str, \langle Str, Str \rangle) \Rightarrow Bool$;
$CreateDerivativeLimitOrderSell(a, m) \triangleq$
  LET $base \triangleq m[1]$
      $quote \triangleq m[2]$
  IN
  $\exists\, quantity, price, margin \in Int :$
    LET $fail \triangleq$
          $\lor\, available[a, base] \leq margin$
          $\lor\, quantity \leq 0$
          $\lor\, price \leq 0$
          $\lor\, \neg IsSellMargin(margin, price, quantity, m)$
    IN
    $\land\, m \in running\_markets$
    $\land\, available' = [available$ EXCEPT $![a, base] = @ - quantity]$
    $\land\, tx' = [type \mapsto$ "create-derivative-limit-order",
            $fail \mapsto fail$,
            $direction \mapsto$ "sell",
                $base \mapsto base,$
                $quote \mapsto quote,$
                $quantity \mapsto quantity,$
                $price \mapsto price,$
                $margin \mapsto margin,$
                $account \mapsto a$
               ]
        $\land\ tx\_fail' = (fail \lor tx\_fail)$
        $\land$ UNCHANGED $\langle balances,\ running\_markets,\ total,\ active\_feeds,\ prices \rangle$

$Next \triangleq$
    $\lor\ \exists\, a \in ACCOUNTS \setminus \{\text{"genesis"}\},\ c \in COINS :$
        $\lor\ Deposit(a,\ c)$
        $\lor\ Withdraw(a,\ c)$
    $\lor\ \exists\, a \in ACCOUNTS \setminus \{\text{"genesis"}\},\ m \in MARKETS :$
        $\lor\ LaunchInstantFuturesMarket(a,\ m)$
        $\lor\ GrantPriceFeeder(a,\ m)$
        $\lor\ RelayPrice(a,\ m)$
    $\lor\ \exists\, a \in ACCOUNTS \setminus \{\text{"genesis"}\},\ m \in MARKETS :$
        $\lor\ CreateDerivativeLimitOrderBuy(a,\ m)$
        $\lor\ CreateDerivativeLimitOrderSell(a,\ m)$

 restrict to non-failing actions only
$NextNoFail \triangleq$
    $Next \land \neg tx\_fail'$
```

```
MODULE MC_futures
EXTENDS FiniteSets, typedefs_futures

ACCOUNTS ≜ {"user1", "user2", "user3"}
COINS ≜ {"inj", "atom"}

 @type: (Str, Str) ⇒ ⟨Str, Str⟩;
pair(i, j) ≜ ⟨i, j⟩

MARKETS ≜ {pair("inj", "atom"), pair("atom", "inj")}

VARIABLES
     cosmos account balances
     @type: ⟨ACCOUNT, COIN⟩ → Int;
    balances,
     available deposits on subaccounts
     @type: ⟨ACCOUNT, COIN⟩ → Int;
    available,
     total deposits on subaccounts
     @type: ⟨ACCOUNT, COIN⟩ → Int;
    total,
     available markets
     @type: Set(MARKET);
    running_markets,
     active price feeds, one per market
     @type: MARKET → Str;
    active_feeds,
     market prices as reported by the price feed
     @type: MARKET → Int;
    prices,
     whether there was a failing transaction
     @type: Bool;
    tx_fail,
     the last executed tx
     @type: TX;
    tx

INSTANCE futures

 @type: Seq(STATE ) ⇒ Bool;
TraceInvFuturesLaunch(hist) ≜
    LET Example ≜
        ∧ ¬hist[Len(hist)].tx_fail
        ∧ ∃ i ∈ DOMAIN hist :
            LET st ≜ hist[i]IN
            st.tx.type = "expiryfuturesmarket-launch"
        ∧ ∃ i ∈ DOMAIN hist :
```

```
        LET $st \triangleq hist[i]$ IN
        $st.tx.type =$ "relay-price-feed"
    IN
    $\neg Example$

@type: $Seq$(STATE ) $\Rightarrow$ $Bool$;
$TraceInvBuySell(hist) \triangleq$
    $\lor$ $hist[Len(hist)].tx\_fail$
    $\lor$ $\neg \exists\, i, j \in$ DOMAIN $hist$ :
        $\land$ $hist[i].tx.type =$ "create-derivative-limit-order"
        $\land$ $hist[j].tx.type =$ "create-derivative-limit-order"
        $\land$ $hist[i].tx.direction =$ "buy"
        $\land$ $hist[j].tx.direction =$ "sell"
        $\land$ $hist[i].tx.account \neq hist[j].tx.account$
        $\land$ $hist[i].tx.base = hist[j].tx.base$
        $\land$ $hist[i].tx.quote = hist[j].tx.quote$
        $\land$ $hist[i].tx.quantity > hist[j].tx.quantity + PRECISION$
        $\land$ $hist[i].tx.price \geq hist[j].tx.price + PRECISION$

$InvBuyForNegativePrice \triangleq$
    $\lor$ $tx\_fail$
    $\lor$ LET $Example \triangleq$
        LET $base \triangleq tx.base$ IN
        LET $quote \triangleq tx.quote$ IN
         $\land$ $tx.type =$ "create-derivative-limit-order"
         $\land$ $tx.direction =$ "buy"
         $\land$ $prices[pair(base, quote)] < -10 * PRECISION$
         $\land$ $tx.quantity \geq 1000$
       IN
       $\neg Example$

@type: $Seq$(STATE ) $\Rightarrow$ $Bool$;
$TraceInvBuySellNegative(hist) \triangleq$
    $\lor$ $hist[Len(hist)].tx\_fail$
    $\lor$ $Len(hist) < 9$
    $\lor$  LET $sell \triangleq hist[Len(hist) - 2]$ IN
       LET $buy \triangleq hist[Len(hist) - 1]$ IN
       LET $hack \triangleq hist[Len(hist)]$ IN
       LET $base \triangleq sell.tx.base$ IN
       LET $quote \triangleq sell.tx.quote$ IN
       LET $max\_price \triangleq -(2^{254} - 1)$ IN
       LET $Example \triangleq$
           $\land$ $buy.tx.type =$ "create-derivative-limit-order"
           $\land$ $sell.tx.type =$ "create-derivative-limit-order"
           $\land$ $buy.tx.direction =$ "buy"
           $\land$ $sell.tx.direction =$ "sell"
```

$$
\begin{array}{ll}
 & \land\ buy.tx.account \neq sell.tx.account \\
 & \land\ buy.tx.base = sell.tx.base \\
 & \land\ buy.tx.quote = sell.tx.quote \\
 & \land\ buy.tx.quantity \geq sell.tx.quantity \\
 & \land\ buy.tx.price \geq sell.tx.price \\
 & \land\ buy.prices[pair(base,\ quote)] = PRECISION \\
 & \land\ sell.prices[pair(base,\ quote)]\ \ = PRECISION \\
 & \land\ hack.prices[pair(base,\ quote)] = max\_price \\
\textsc{in} & \neg Example
\end{array}
$$

use this view to enumerate various scenarios

$$
TxView \triangleq \\
\quad \langle tx.type,\ tx\_fail \rangle
$$

MODULE *typedefs_futures*

```
@typeAlias: ACCOUNT = Str;
@typeAlias: COIN = Str;
@typeAlias: BALANCE = <<ACCOUNT, COIN>> -> Int;
@typeAlias: MARKET = <<COIN, COIN>>;
@typeAlias: TX = [ type: Str, fail: Bool, account: ACCOUNT, coin: COIN,
                   base: COIN, quote: COIN, quantity: Int, price: Int,
                   direction: Str, margin: Int, status: Str ];
@typeAlias: STATE = [ balances: BALANCE, available: BALANCE, total: BALANCE,
                      tx: TX, running_markets: Set(MARKET),
                      active_feeds: MARKET -> Str, prices: MARKET -> Int,
                      tx_fail: Bool ];

a dummy definition to define aliases
typedefs_aliases ≜ FALSE
```