OneDro1d Argus

CASE STUDY

Does the guardrail hold on every path? Seven sealed checks against a FINOS CALM demo

One sealed set of checks, run with OneDroid Argus against the FINOS CALM "trades API and MCP" demo: whether the running system matches its architecture, whether its guardrail holds on every path, and how anyone can check the result later.

Case study. Certified run of 7 October 2026. OneDroid Argus, one system, in early use.

In short

- Our team ran one sealed set of seven checks with OneDroid Argus against the FINOS CALM "trades API and MCP" demo: a trades REST API, an MCP server for an AI agent with a guardrail on restricted symbols, and an A2A server.
- Six checks were generated from the demo's own CALM architecture file and one was written by hand. OneDroid Argus reported 5 passed, 2 failed and 0 errored.
- The guardrail refused the restricted symbols in a trade search, but returned a restricted trade in full when the MCP server was asked for it by id.
- A connection the architecture forbids was open. The cause is our cluster: it does not enforce Kubernetes network policies, so the demo's own policies had no effect there.
- The checks were sealed and recorded on a public test network and in Bitcoin before the run, and the verdict after it. Before running, OneDroid Argus confirmed that the images running were exactly the ones declared.

The question

A team is about to put an AI agent's tool server in front of trading data. Its architecture says what may talk to what, and a guardrail says which symbols the agent must never see. Before trusting it, the team wants to know three things: does the running system match the architecture, does the guardrail hold on every path, and can anyone prove later exactly what was tested and what it found?

FINOS CALM is an open source way to write a system's architecture down as a file: its parts, which of them may talk to which, and the controls on them. The demo tested here is the "trades API and MCP" scenario (QCon scenario 2). This is not a review of FINOS CALM or of the demo, and not a recommendation. It is a record of what one set of checks saw.

The setup

Item	Value
System	The FINOS CALM "trades API and MCP" demo: a Trades REST API, a Trades MCP server with an `mcp-guardrail` control (denied symbols VOD, GME and AMC), and an A2A server
Images	The demo's public images, pinned by digest: trades-rest-server `sha256:21bc0b16…`, trades-mcp-server `sha256:feb820a3…`, trades-a2a-server `sha256:2ec2a457…`
How it ran	The demo's own manifests, unchanged, on a shared Azure Kubernetes cluster in Poland Central. The demo is stateless and generates its trade data in memory when it starts
OneDroid Argus	Executor v0.3.66 on the same cluster, with read-only access to the demo's pods so that a certificate can say what was running. One system, no reference build
The checks	6 generated by `argus calm import` from the demo's CALM architecture file, 1 written by hand
Mode	Certifying (`final`): the set is sealed and recorded before the run
Who wrote and ran the checks	Our team

What was checked

The certified set holds seven checks, sealed on 7 October 2026 at 19:57 UTC.

Check	Where it comes from	What healthy means
The MCP client reaches the MCP server	The architecture declares this connection	It answers
The MCP client path cannot reach the Trades API	The architecture declares no such connection	No answer: the path is closed
The guardrail lets an allowed symbol through	The guardrail control	A search for LSE:AAPL returns trades
The guardrail refuses VOD on the trade search	The guardrail control	Refused
The guardrail refuses GME on the trade search	The guardrail control	Refused
The guardrail refuses AMC on the trade search	The guardrail control	Refused
A trade in a denied symbol, asked for by id, is refused	Written by hand. The id is looked up at run time from the system of record	Refused

The first six came straight from the architecture file: a connection that must work, a connection that must not, the guardrail's allow and its three denies. The seventh asks the question the architecture file does not: whether the guardrail also holds on a second way to the same data.

The result

	Certified run, 7 October 2026
Checks	7
Passed	5
Failed	2
Errored	0
Run status	failed

The run status is the word OneDroid Argus gives a run. It reads `failed` when at least one check failed, and it is not a judgement of the product: the counts above are the result. An errored check is one that could not run, and an error is never counted as a pass.

No check was left not measured, and none could not run. The numbers are the ones OneDroid Argus reported. OneDroid Argus gives no score. An earlier certified run of the same seven checks the same day gave the same verdict.

The two checks that failed

1. The guardrail does not cover lookup by id. The trade search refuses VOD with "is restricted". Asking the MCP server for a VOD trade by its id returned the full trade: instrument, price, quantity and status. It was seen in every run that day in which the trade asked for was a VOD trade, including both certified runs. We did not trace the cause in the demo's code, and we have not reported it to the demo's authors.

2. A connection the architecture forbids was open. The architecture declares no connection from the MCP client to the Trades API. A request on that path got an HTTP answer (404), so the network let it through. It was seen in every final run that day. The cause is traced, and it is ours: the cluster we ran on does not enforce Kubernetes network policies, so the demo's own policies had no effect there. This is about our deployment, not the demo's code, and it may not appear where policies are enforced.

The second finding is still the useful one. An architecture file says what must not connect. Whether the network underneath agrees is a property of each deployment, and a check generated from the file asks it every time.

What passed

- **The declared connection.** The MCP client reached the MCP server.
- **The guardrail on search.** It refused VOD, GME and AMC on the trade search, and let LSE:AAPL through.
- All of this held in every run.

How anyone can check it later

Before the run, the sealed set was recorded on three chains: a private chain run by OneDroid Argus, Base Sepolia (a public Ethereum test network) and Bitcoin, through OpenTimestamps. After the run, the verdict was recorded on the same three. What goes on a chain is only ids, hashes, digests, counts and the verdict, never a check, an expected value or an observed one.

Record	Base Sepolia transaction
The sealed set, before the run (block 47817413)	`0x9fccbd30504e7e31554455dfc27bbf27ca5cba4cd5e30d9f607a6355aea32adc`
The verdict, after the run (block 47817439)	`0x8718c0b64c7fd35feb06029e1af1574b944d35c2afde6676a114132b7353ce7a`

Each transaction can be opened on sepolia.basescan.org. The Bitcoin record of this run was submitted to OpenTimestamps, and its Bitcoin attestation lands within hours of the run. A certificate built from these records can be checked with the `argus` command line, with no account and without seeing a single check: docs.onedroid.ai/argus-ledger.

What was running was measured, not assumed. Before the run, OneDroid Argus read the image digests actually running and found exactly the three that the certification declared. To show the other side, a final run was started that declared a digest that was not running. It was refused before

any check ran: "is not among the image digests running in the system under test — no scenario was run and nothing was bound".

The seal proves priority: the checks existed, unchanged, before the result. It does not make their authors blind to the system. The same team wrote the checks and ran them.

What this test did not cover

Not covered	Why it matters
More than one deployment	One shared cluster without network policy enforcement. The second finding may not appear where policies are enforced
More than one certified run of this set	One certified run, plus one earlier certified run of the same checks the same day. Two runs, not a stability measure
Real data	In-memory demo data, regenerated at each start
Capacity	Not certified. Two build runs, outside the sealed set, served up to 160 concurrent users with a 95th percentile of 106 ms or less and no errors. The limit was not reached in this setup, so there is no capacity claim
Browsers and user interfaces	HTTP and MCP checks only
The A2A server	No check in the set asked it a question
The cause of the first finding	Not traced in the demo's code
An independent tester	The same team wrote the checks (six generated from the architecture, one by hand), ran them and wrote this, and OneDroid Argus is our product
A released product	OneDroid Argus ran here against a development control plane

What a run on one system can and cannot say

It can say	It cannot say
Whether the running system keeps each connection the architecture declares, and blocks each one it does not	Whether the architecture itself is a good one
Whether a control holds on every path a check asks about	Whether it holds on a path no check asks about
Which images were running when the verdict was given	Anything about a deployment it did not run on
That the checks were fixed before the result, and the result not changed since	That nobody looked at the system first

What we take from it

- An architecture file is a list of claims. Six of the seven checks were generated from it, so the claims were tested as written, not as remembered.
- A guardrail on one path is not a guardrail on the data. The one hand-written check, a second way to the same trade, is the one that found the gap.
- A rule about what must not connect depends on the network under it. Check it on every deployment, including your own.
- A run status of `failed` is one word. The table under it is the result, and the chain records let anyone confirm it was not changed afterwards.

The video: https://www.youtube.com/watch?v=V6SsYw3b9bQ

FINOS and CALM are names that belong to their respective owners, as do QCon, Kubernetes, Azure, Bitcoin, Base and OpenTimestamps. OneDroid is not affiliated with FINOS, the CALM project or the demo's authors, and the names are used only to identify the software that was tested and the networks the records are on. OneDroid Argus is built by Providentia Worldwide. Questions: michal@onedroid.ai. How OneDroid Argus works: onedroid.ai/argus.