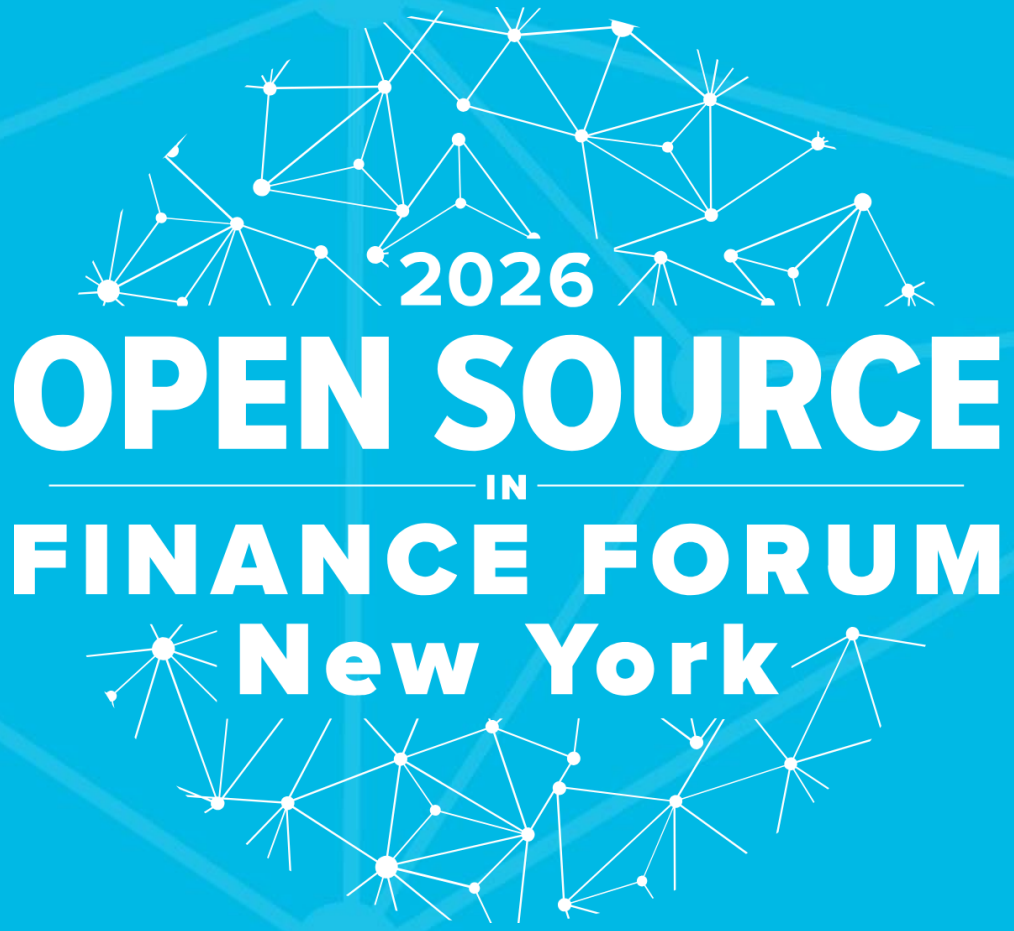


The Last Mile of Cloud Controls

Using AI to Operationalize FINOS CCC

Maxime Coquerel

Disclaimer

“Any views or opinions expressed in this presentation are those of the presenter and do not necessarily represent the views and opinions of his employer, its ownership, management or employees.”

About me

Maxime Coquerel

Principal Cloud Security Architect – RBC
CNCF TAG Security & Compliance – Tech Lead
CNCF Ambassador
Contributor – FINOS Common Cloud Controls

Email: max.coquerel@gmail.com
Blog: zigmax.net
GitHub: github.com/zigmax
X: @zig_max



FINOS CCC: One Shared Rulebook for Cloud Security

FINOS COMMON CLOUD CONTROLS

An open standard: one unified set of cybersecurity, resiliency and compliance controls for common services across the major cloud providers.

- Hosted by FINOS under the Linux Foundation
- Written by banks, cloud providers and the community
- Published as versioned YAML on GitHub and ccc.finos.org
- Steering committee: BlackRock, Citi, LSEG, Morgan Stanley, RBC

458 capabilities

181 threats

167 controls

Every CCC Check Traces Back to a Threat

Capability

What can the service do?

A managed Kubernetes control plane with an API

`CCC.K8S.CP01`

Threat

What could go wrong?

The API is reachable from untrusted networks

`CCC.K8S.TH01`

Control

What must be in place?

Restrict network access to the API

`CCC.K8S.CN01`

Requirement

How do we check it?

Public access to the API endpoint MUST be disabled

`CN01.AR02`

CCC.Core

Shared entries every service needs: encryption, MFA, logging.

Service catalogs

One per cloud service type, e.g. CCC.K8S for Kubernetes. They reuse Core.

Every link is an ID, so software can follow the chain from a service to its checks.

Standard ≠ Internal Policy ≠ Proof

EXAMPLE · KUBERNETES PRIVATE CLUSTER

CCC REQUIREMENT · PUBLIC

CCC.K8S.CN01.AR02

When a Kubernetes API endpoint is active, public network access to that endpoint MUST be disabled.

INTERNAL REQUIREMENT · PRIVATE

INT-NET-02

Approved application segmentation: no management plane exposed to the internet.

CLOUD IMPLEMENTATION · AZURE AKS

Private cluster

`apiServerAccessProfile`

```
enablePrivateCluster
  = true
enablePrivateClusterPublicFQDN
  = false
```

Public control intent ≠ internal policy ≠ technical implementation

The setting is a proposed mechanism. Proof comes from the deployed configuration and a negative test.

Six Terms, in Plain Words

PUBLIC · FROM CCC

Requirement

A testable sentence from CCC. It says what MUST be true.

`CCC.K8S.CN01.AR02`
public API access MUST be disabled

PRIVATE · THE BANK

Internal policy

The bank's own rule, with its own ID and an owner. Never published.

`INT-NET-02` · owner: Network security

THE LINK

Mapping

Which internal rule matches which CCC requirement. Exact, or only partial.

CCC TLS 1.3 ↔ internal TLS 1.2: partial

PROOF

Evidence

What shows the rule is really in place. A claim in a document is not evidence.

`az aks show` + a public access test that must fail

TRACEABILITY

Pinned

A fixed version of CCC, checked by its fingerprint (SHA-256). The review can be replayed.

CCC.Core `v2025.10` · `86b7cc65…`

AUTOMATION

Policy-as-code

A rule a machine enforces on every deployment, not just once in a review.

Deny AKS clusters where `enablePrivateCluster` is not true

The Last Mile Connects Three Worlds

OPEN STANDARD

Consistent control intent, shared vocabulary, versioned capabilities, threats, controls and requirements.

REAL ARCHITECTURE

Services, trust boundaries, identities, connectivity, data flows, deployment choices and unknowns.

PRODUCTION DECISION

Private internal requirements, implementation proof, exceptions, residual risk and accountable approval.

The last mile is a traceable translation between these three worlds.

Today, a Review Looks Like This

A MANUAL REVIEW TODAY · ILLUSTRATIVE

Payments API on AKS

- Is the cluster private? Check ingress justification.
- TLS? Probably fine on App Gateway.
- Pod security not mentioned.
- Data says Confidential. Cardholder data?

Good instincts, from an experienced reviewer.

WHAT IS MISSING

- ✖ Which control? No ID
- ✖ Which internal rule? None
- ✖ Who owns it? No one
- ✖ What proof to ask for? Vague
- ✖ Which version of the standard? Unknown

Good instincts, no trace. Next review, another reviewer, other notes.

Writing the Rules Is Solved. Applying Them Is Hard.

1

Two languages

The public standard and the bank's internal policy say similar things, with different words, IDs and owners.

IN THE DEMO

```
CCC.K8S.CN01.AR02
↕ translated by hand
INT-NET-02
```

2

A claim is not proof

“Our cluster is private” is a statement. Proof is the deployed configuration and a test that fails from the internet.

IN THE DEMO

```
az aks show
+ public connectivity test
(must fail)
```

3

Rules don't always match

The standard and internal policy can disagree. No tool can settle that: someone accountable must decide.

IN THE DEMO

```
CCC: TLS 1.3
Internal: TLS 1.2
Which bar applies?
```

Three reasons the translation is still done by hand, one review at a time.

Why Not Just Ask an LLM?

RISK	WHAT IT LOOKS LIKE	SAFEGUARD IN THIS PATTERN
It invents	Cites a CCC control that does not exist →	ID and text must match the pinned CCC files
It assumes	“TLS is probably fine” becomes a fact →	Basis must be quoted word for word from the input
It concludes	Declares the control “passed” →	It may only write “unverified”. A human decides.
It exposes	Needs the bank's confidential policies →	In this prototype, the AI never sees `internal-mapping.json`

AI speeds up the translation **Code checks it** **A human decides**

The Answer: AI Proposes, Code Checks, a Human Decides

1 AI proposes

AI · SKILL.MD

Picks the CCC controls that apply, explains why in its own words, and asks one precise question each.

uses `architecture.json` `controls.json`

→ **status is always "unverified"**

2 Code checks

CODE · REVIEW.PY

Real CCC ID? Basis quoted word for word? Not "passed"? No owner invented? Otherwise: rejected.

uses `ccc/` pinned by SHA-256

→ **4 bad proposals rejected**

3 Code adds private rules

CODE · REVIEW.PY

Joins each internal requirement and its owner, from a file the AI never sees. Flags "partial" mappings.

uses `internal-mapping.json` **private**

→ **TLS: internal 1.2 vs CCC 1.3**

4 Traceable draft

OUTPUT · RUN 1

3 controls · 3 evidence missing · 1 ambiguous mapping. Each finding: threat, CCC, internal rule, owner, evidence.

→ **review-run1.md**

5 Team sends evidence

APP TEAM · RUN 2

Answers and files come back. 2 controls move to "review pending". 1 is still missing.

uses `evidence/answers.json`

→ **review-run2.md**

6 Human decides

HUMAN · RISK OWNER

TLS evidence shows 1.2: meets internal policy, not CCC. Upgrade, accept, or record an exception?

→ **never "passed"**

AI never sees private rules **Code never decides** **Only a human accepts risk**

The Input: An App With Three Unknowns

architecture.json

```
{
  "_about": "Fictional application for the demo. Not an RBC system.",
  "name": "Customer Payments API",
  "data": "Confidential (claimed by the team, not verified)",
  "components": {
    "Azure Kubernetes Service": [
      "Hosts the payments API",
      "API server access: unspecified",
      "Pod security: unspecified"
    ],
    "Azure Application Gateway": [
      "Internet-facing, routes customers to AKS",
      "TLS minimum version: unspecified"
    ],
    "Azure SQL Database": [
      "Stores payment transactions"
    ]
  },
  "evidence": []
}
```

KNOWN

AKS, App Gateway (internet-facing), Azure SQL. Data: Confidential (claimed).

UNKNOWN

API server access, TLS minimum version, pod security.

EVIDENCE

None supplied yet.

Can we approve this from a description alone? No.

What the AI Adds: Select, Explain, Ask

1 Select

4 CCC REQUIREMENTS IN · 3 APPLY

- ✔ **`CCC.K8S.CN01.AR02`**
  API server access: unspecified
- ✔ **`CCC.Core.CN01.AR01`**
  TLS minimum version: unspecified
- ✔ **`CCC.K8S.CN05.AR02`**
  Pod security: unspecified
- ✖ **`CCC.Core.CN01.AR02 · SSH`**
  Not applicable: no component exposes SSH

→

2 Explain

WHY TLS APPLIES · THE AI'S OWN WORDS

> "The Application Gateway is internet-facing and carries payment traffic, so it exposes a non-SSH port. This requirement needs TLS 1.3 or higher there, and the minimum version is not stated."

Grounded in: `"TLS minimum version: unspecified"`

→

3 Ask

ONE PRECISE QUESTION

> **"Does the Application Gateway SSL policy set TLS 1.3 as the minimum?"**

A question the team can answer with evidence, not "is it secure?".

Status: **unverified**.

This is judgment, not lookup. **Next: the code checks every word of it.**

The Code Checks Every Proposal

① CONTROL IDENTIFIED BY THE AI

`CCC.Core.CN01.AR01`

TLS 1.3 or higher on the internet-facing gateway

Owner, added by code: **Platform engineering**

+ 2 more controls accepted
+ 1 judged not applicable (SSH)

② WHAT THE VALIDATOR SAID

3 accepted

4 rejected

Accepted: real CCC ID, exact quote, only “unverified”, no invented owner.

Rejected: 4 mistakes planted on purpose. Real terminal output: next slide.

③ DECISION

Human review required

The TLS mapping is only **partial**: internal policy accepts 1.2, CCC requires 1.3.

Never “passed”.

AI proposes **Code checks** **A human decides**

The Real Output: Run 1

bash: ccc-osff-nyc-26 — REAL OUTPUT

```
~/ccc-osff-nyc-26 $ python3 review.py
(no proposals.json yet: using the recorded skill output)
3 controls | 3 evidence missing | 0 review pending | 1 ambiguous mapping | 0 rejected | 1 judged not applicable
  AMBIGUOUS CCC.Core.CN01.AR01 -> INT-CRY-05: Internal floor is TLS 1.2; CCC requires TLS 1.3. Meeting INT-CRY-05 does not meet CCC.
Decision: HUMAN REVIEW REQUIRED -> review-run1.md

~/ccc-osff-nyc-26 $ python3 review.py --proposals examples/bad-proposals.json
0 controls | 0 evidence missing | 0 review pending | 0 ambiguous mapping | 4 rejected | 0 judged not applicable
  REJECTED CCC.K8S.CN01.AR09: ID not in the pinned CCC catalog
  REJECTED CCC.Core.CN01.AR01: basis is not quoted from the architecture
  REJECTED CCC.K8S.CN05.AR02: status 'passed' not allowed
  REJECTED CCC.K8S.CN01.AR02: the model may not supply the internal mapping or owner
Decision: HUMAN REVIEW REQUIRED -> review-bad-proposals.md
```

review-run1.md: generated report (top) — REAL FILE

Control review draft: Customer Payments API (run1)

Decision: Human review required. This tool never marks a control as passed.

- Candidate controls: 3
- Evidence missing: 3
- Review pending: 0
- Ambiguous mappings: 1
- Judged not applicable by the AI: 1
- Rejected proposals: 0

Pinned: CCC.Core v2025.10 (controls `86b7cc6592d5`, threats `bf733e8a05ff`)

Pinned: CCC.K8S main@1188050 (unreleased) (controls `874823161c62`, threats `e32f05264c6e`)

Policy Says Yes, CCC Says No: A Human Decides

THE TEAM'S EVIDENCE

`appgw-ssl-policy.json`

App Gateway minimum

TLS 1.2

A real file. But is it enough?

→

TWO RULES, TWO ANSWERS

Bank policy `INT-CRY-05`
requires TLS 1.2 or higher
✔ Met

FINOS CCC `CCC.Core.CN01.AR01`
requires TLS 1.3 or higher
× Not met

review.py flags the mapping as **partial**. It does not pick a winner.

→

THE RISK OWNER DECIDES

Which bar applies?

- Upgrade to TLS 1.3
- Accept TLS 1.2 + record an exception

Status: review pending. Never "passed".

INT-CRY-05 accepts TLS 1.2, CCC requires 1.3. Status: "review pending", never "passed".

The Real Output: Run 2

bash: ccc-osff-nyc-26 — REAL OUTPUT

```
~/ccc-osff-nyc-26 $ python3 review.py --evidence evidence/answers.json
(no proposals.json yet: using the recorded skill output)
3 controls | 1 evidence missing | 2 review pending | 1 ambiguous mapping | 0 rejected | 1 judged not applicable
  AMBIGUOUS CCC.Core.CN01.AR01 -> INT-CRY-05: Internal floor is TLS 1.2; CCC requires TLS 1.3. Meeting INT-CRY-05 does not meet CCC.
Decision: HUMAN REVIEW REQUIRED -> review-run2.md
```

review-run2.md: generated report, TLS finding — REAL FILE

CCC.Core.CN01.AR01: Encrypt Data for Transmission

- **Architecture fact (unverified):** "TLS minimum version: unspecified"
- **AI reasoning:** The Application Gateway is internet-facing and carries payment traffic, so it exposes a non-SSH port. This requirement needs TLS 1.3 or higher there, and the minimum version is not stated.
- **CCC threat:** CCC.Core.TH02 Data is Intercepted in Transit
- **CCC requirement:** When a port is exposed for non-SSH network traffic, all traffic MUST include a TLS handshake AND be encrypted using TLS 1.3 or higher.
- **Internal requirement (private):** INT-CRY-05 Minimum protocol versions in transit: TLS 1.2 or higher
- **Mapping: PARTIAL.** Internal floor is TLS 1.2; CCC requires TLS 1.3. Meeting INT-CRY-05 does not meet CCC.
- **Owner:** Platform engineering
- **Question:** Does the Application Gateway SSL policy set TLS 1.3 as the minimum?
- **Evidence request:** Application Gateway SSL policy (minimum protocol version); handshake test with a TLS 1.2 client (must fail)
- **Policy-as-code candidate:** Deny Application Gateways whose SSL policy allows a minimum below TLS 1.3
- **Team answer (a claim, not proof):** App Gateway uses the predefined policy AppGwSslPolicy20220101S.
- **Evidence file:** `evidence/appgw-ssl-policy.json`
- **Status:** review pending

appgw-ssl-policy.json — EVIDENCE


```
{"_synthetic": "az network application-gateway
ssl-policy show",
 "policyType": "Predefined", "policyName":
"AppGwSslPolicy20220101S", "minProtocolVersion":
"TLSv1_2"}
```

Before and After: Same Concern, Now Traceable

What the reviewer needs	Manual notes	Skill output
Control reference	None	CCC.K8S.CN01.AR02 + TH01
Internal requirement	None	INT-NET-02 Approved segmentation
Accountable owner	None	Network security
Evidence request	“Is the cluster private? Check ingress justification.”	az aks show (apiServerAccessProfile), public connectivity test (must fail)
Policy-as-code	None	Deny AKS clusters without enablePrivateCluster
CCC version checked	Unknown	CCC.Core v2025.10, K8S main@1188050

Same concern. The difference is traceability.

What AI Speeds Up, and What It Can't Certify

HIGH-VALUE ASSISTANCE

- Fast retrieval and coverage discovery
- Candidate CCC ↔ internal mappings
- Consistent drafting and questions
- Evidence checklist creation
- Policy-as-code test scaffolding

HUMAN OR DETERMINISTIC GATES

- Applicability to business context
- Correctness of private mappings
- Effective configuration testing
- Exceptions and compensating controls
- Residual risk acceptance and approval

Model confidence is not control effectiveness.

How Do We Know the AI Is Right? Measure It.

Indicator	Question
Control coverage	What share of the relevant controls was identified?
Mapping accuracy	How many CCC → internal mappings are correct?
Citation validity	Does every cited control exist in the pinned release?
Evidence quality	Is the requested evidence sufficient for the control?

WATCH FOR Omissions Hallucinated IDs False positives Wrong applicability

Score it against a reference set labelled by security architects, over repeated runs.

Three Takeaways

01

Open standard

Use FINOS CCC as the shared, versioned vocabulary for cloud control intent.

02

Checked AI

Let AI draft from pinned CCC; code checks every proposal and adds private rules.

03

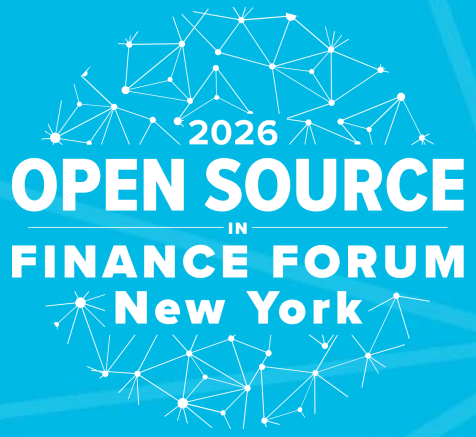
Human ownership

Keep validation, exceptions and risk acceptance outside the model.

Standards define what good looks like. AI helps us operationalize them.

Evidence proves whether controls work.

FINOS CCC provides the common language. AI helps bridge the last mile. Humans remain accountable for the decision.



Thank You · Questions?

From open control intent to evidence-backed cloud decisions.

Maxime Coquerel · RBC

Reproduce it: SKILL.md · synthetic architecture · runner · report → github.com/zigmax/ccc-osff-nyc-26

`ccc.finos.org · github.com/finos/common-cloud-controls · calendar.finos.org`