woven BY TOYOTA

Building an AI for Security Exception Tickets: What Actually Worked

September 10th, 2026

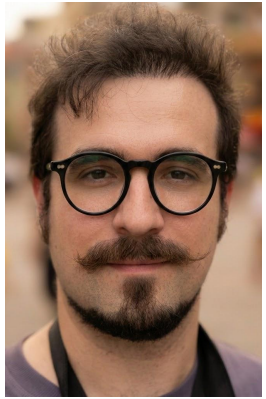
Juan Corena
Senior Software Engineer
juan.corena@woven.toyota

Index

The team

Juan Corena

Senior Software Engineer



Goktug Serez

Senior Enterprise Security Engineer

John-Carlo Pineda

Enterprise Security Engineer

Jacob Ceresia

Technical Program Manager

Problem

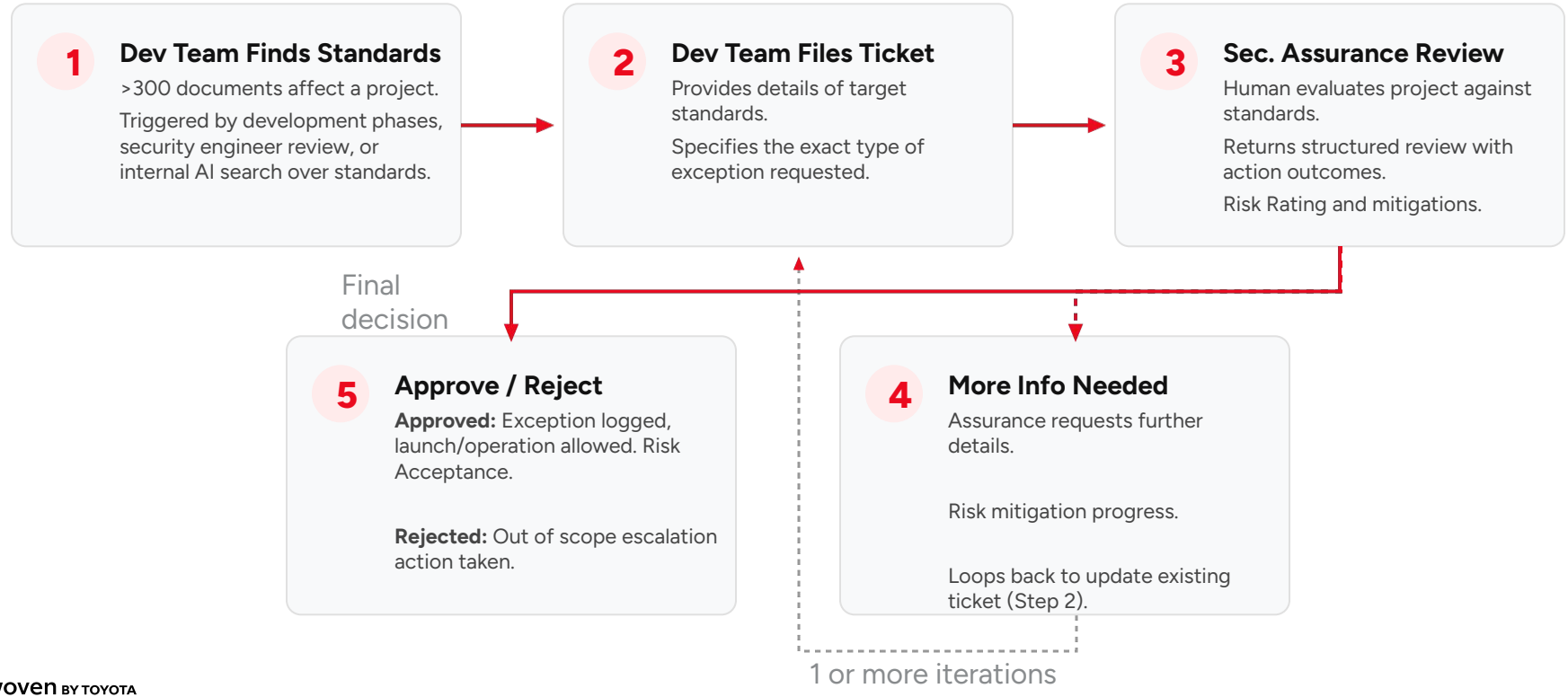
Some teams need exceptions on some of the internal security standards.

Exception granting is complex

Type	Count (Magnitude)
Standards	Hundreds
Processes	Dozens
Policies	Dozens
Procedures	Dozens
Guidelines	Dozens
Total	More than 300 docs

Existing Approach

Original Workflow



Challenges

Many sec. standards

Many security standards make it difficult for users to know every detail of them.

Standard needs to balance security, completeness and usability.

Range of projects

From automotive to entire cities.

This makes work harder for Security Assurance team members.

Review time

Needs to be balanced with other tasks within the Security Assurance team.

Idle time

Back and forth between different time zones slows down the procedures.

Days can be wasted due to time differences between Tokyo, London and Palo Alto.

~ 1 week

Challenges (2)

Manual selection can lead to incompleteness

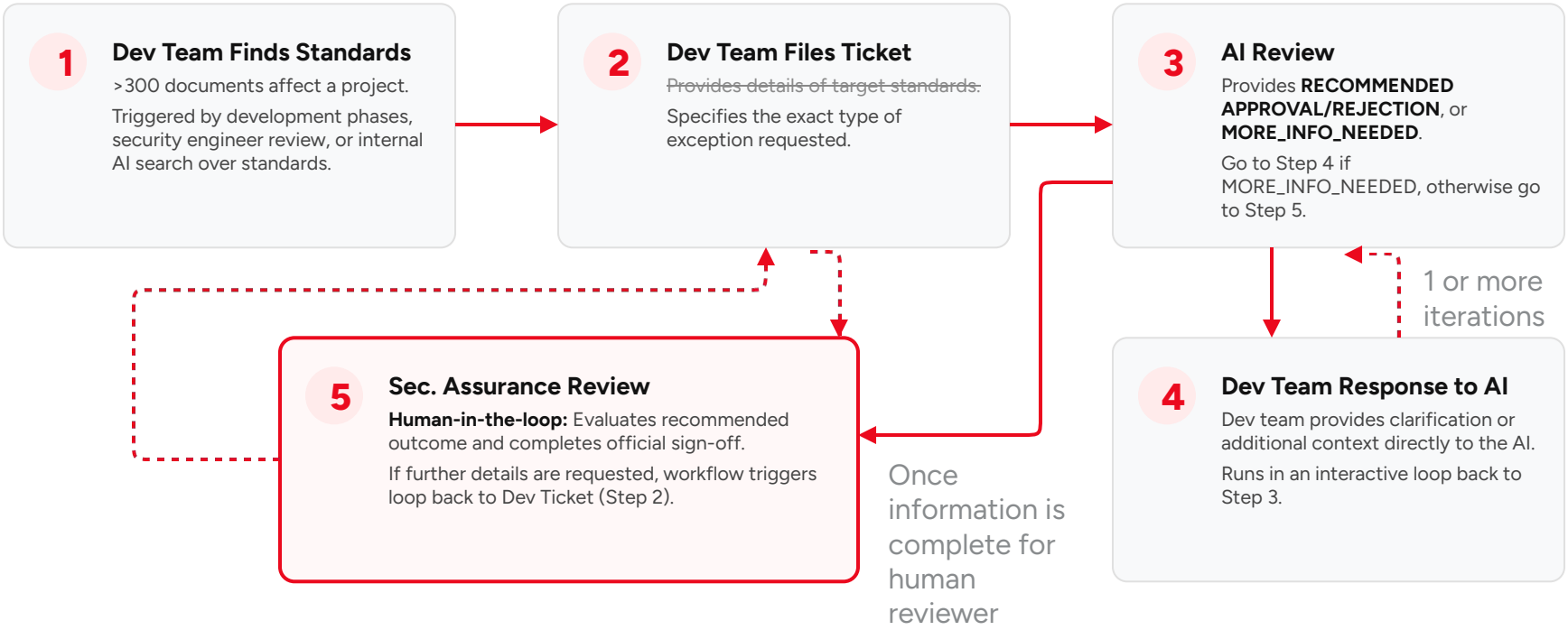
In the original process, those asking for the exception might not be aware or all the relevant standards.. For this reason, tickets could be incomplete at filing time.

Humans are still needed

- Logically speaking this is about **exceptions,** even a perfect deduction system would just deny every exception.
- Some projects involve critical software.
- Trade-offs need to fall on a human person.
- More than one human expert needs to be involved (e.g. legal, vendor, amongst others).

Using AI to improve the process

AI-assisted Workflow



Our Infra and Input

Security Standards at Woven by Toyota

Our standards can be easily consumed by LLMs.

- **All in Markdown format**
- All live in the same code repository with CI/CD (updates info).
- Company wide access to LLM infrastructure with several models.
- Coding agents are also available.
- Allowed local open models (prevents vendor lock-in).
- Ticketing system has API (retrieve/post tickets).
- Internal messaging tool allows bots (display results).

From an LLM perspective

- Roughly 800k tokens worth of text to process.
- About 3k tokens per standard (6 or 7 pages).
- Context is not huge, but we need to be careful to use local models.

1 - Synchronizing context

Design decisions to keep sources updated

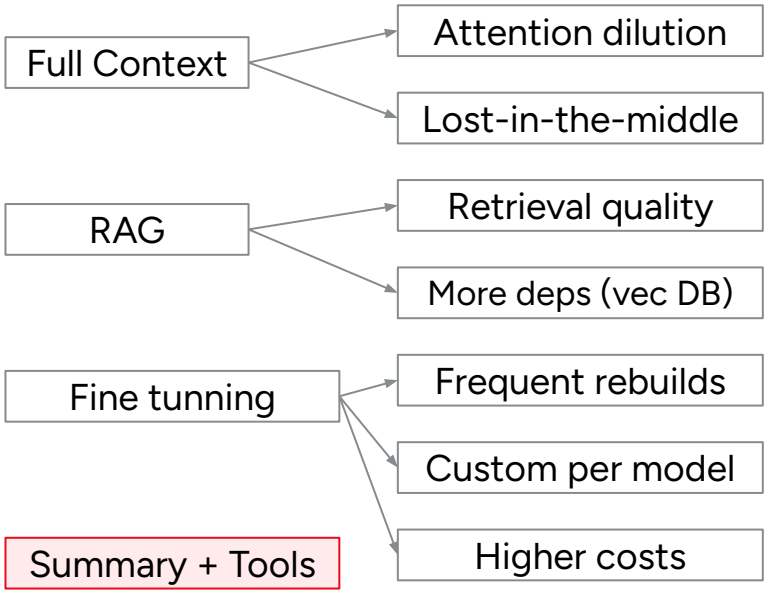
- Standards are modified on average every 1.5 days (PRs / day)
- A daily job was created to keep the data sources updated
- CI gets diffed files since last successful run of current workflow
- Files in the right directories (e.g. standards) are summarized
- The Process produces a file (see right)
- The file is uploaded as a CI artifact


```
{
 "standards": [
  {
   "id": "standard-1",
   "summary": "The document outlines the
                    Application..."
   "contents": "---\ntitle: Application ..."
  },
 ...
}
```

1 - Synchronizing context

Why this output format?

- We wanted something that worked.
- Context is big enough to require tuning for retrieval.
- My team is not an AI department, we wanted to **minimize the cognitive load** for the maintenance of this project.
- Increasing context, gave good results, we used that.



2 - Exposing the context

We used MCP exposing two functions

- *Full list of summaries* with ids (e.g. "standard-1") and *Retrieve full context by id*.
- This reduces the context a lot, most requests are about a single standard (e.g. Cryptography).
- Why not a Claude/Copilot/Codex skill? Open source libraries provide **more flexibility and control** , standards **can be exposed through a public protocol** for the entire company to use.
- Any drawbacks? MCP uses more tokens than just reading the files, but it helps control the data source better.

1. Analyze Ticket

The agent processes and analyzes the incoming security exception ticket to understand the core context.

2. Check Summaries

Queries the MCP tool to scan the full list of standard summaries and identify matching criteria.

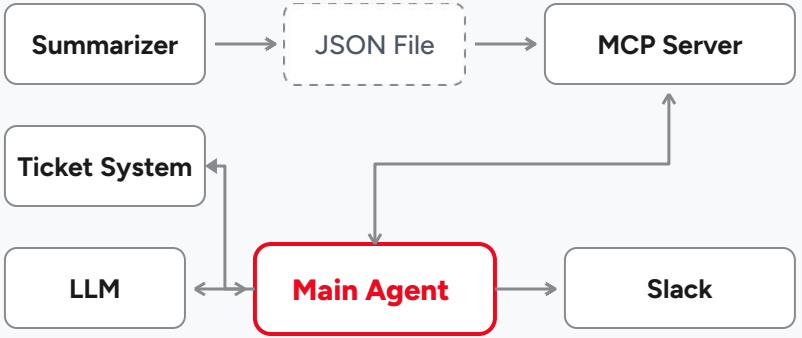
3. Retrieve Only Needed Standards

Fetches exclusively the full text of the target standards, minimizing total token usage and overhead.

3 - Executing the agent

We used Golang for the main agent, summarizer, mcp-server

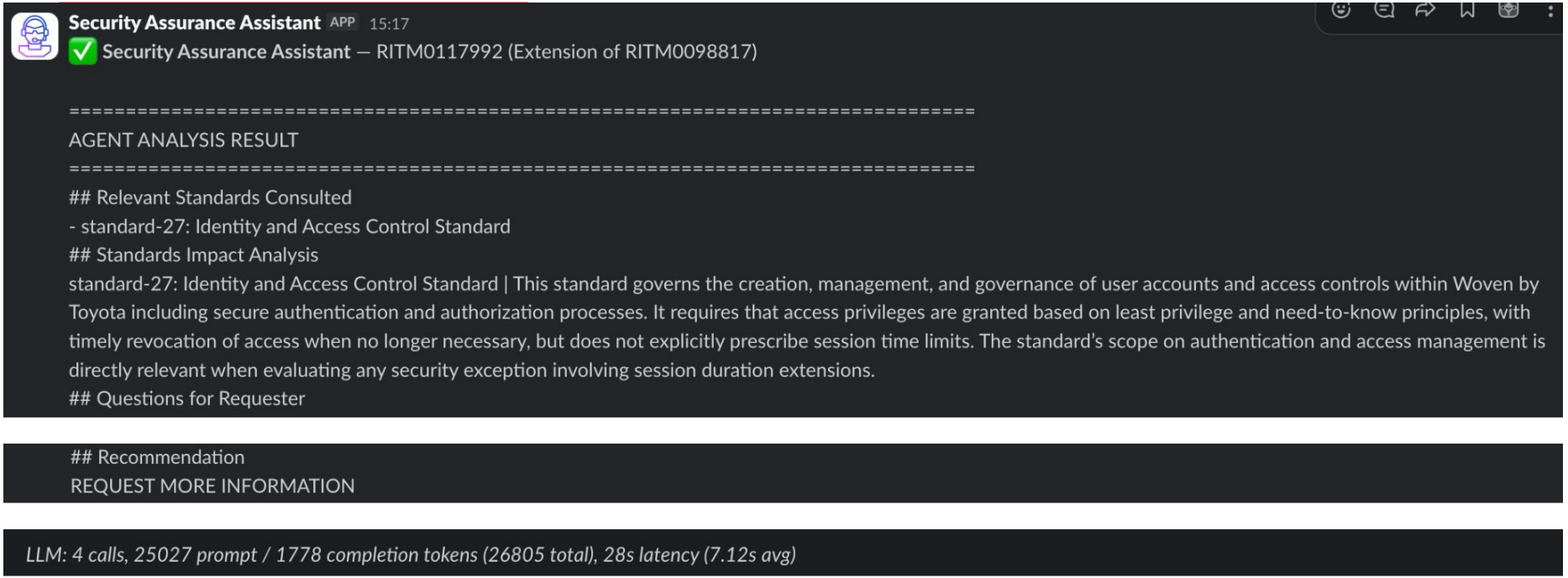
- The agent triggers daily, connects to the Ticketing system.
- Used Google's adk-go framework.
- We prefer compiled languages for production software: speeds up the coding (feedback loop), reduces runtime deps.
- Using an agent framework helped us avoid reinventing the wheel.
- Go has a good standard library, reduces maintenance time by not having too many external libraries.
- MCP protocol used the official MCP sdk for Go.



Summarizer: Once a day workflow.

Main Agent: Orchestrates the execution flow and relays outputs to communication channels.

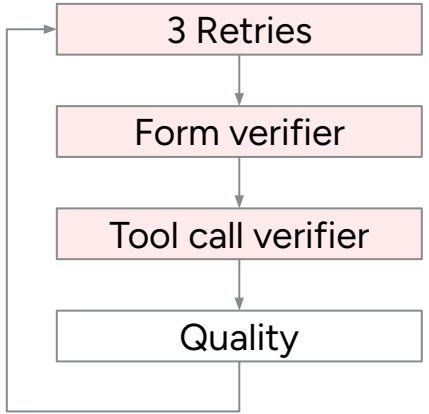
3 - Executing the agent



4 - Agent reliability

The goal was not to waste human time

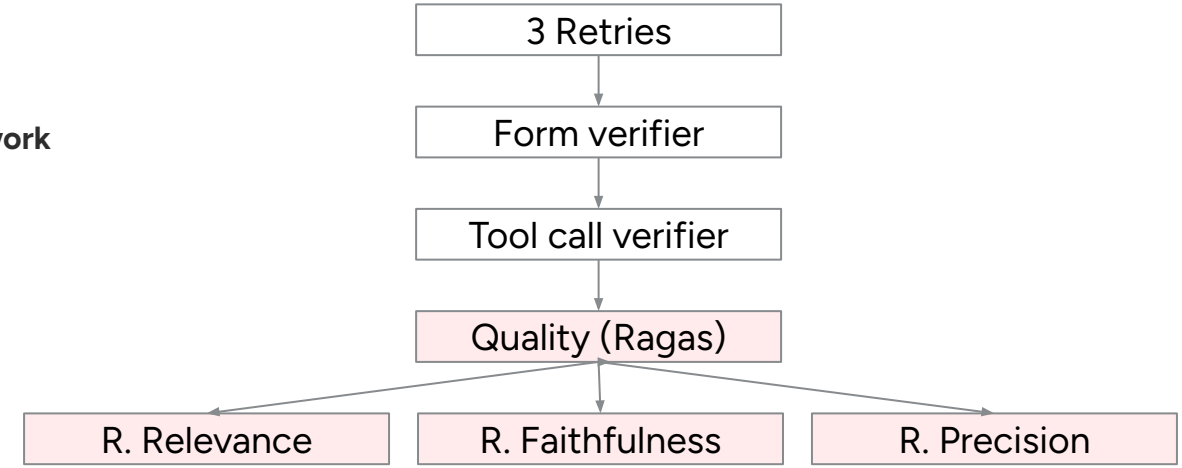
- 3 retries per ticket (1st: 86%, 2nd: 99%).
- Verify form: 1 >= standard used, text for reasoning, questions when needed, recommendation not empty (e.g. approve), reasoning >= 100.
- Verify tool calls: MCP calls were done, standard ids were retrieved.



5 - Output quality

Correctness + Ragas (R.) evaluation framework

- Coherence: does the final **recommendation** (e.g. approve), follow from the **reasoning and question.**
- Specificity: Is the answer specific to this case
- Relevancy: does the answer address the question?
- Faithfulness: are the answer **claims** based on **retrieved context**?
- Precision: was the context useful for answering?



5 - Output quality - Metric Justification (food edition)

Coherence

Fact: mustard is required on all hot dogs
Question: hot dog has no mustard
Reasoning: needs mustard, but this…
Recommendation: approve compliant hot dog

Contradictory recommendation

Specificity

Fact: mustard is required on all hot dogs
Question: no mustard in stock, please allow ketchup at dinner event serving hot dogs
Recommendation: Follow condiment rules.

Generic. ketchup ok, get mustard or no hot dogs.

Faithfulness

Fact: only mustard is allowed on hot dogs
Hallucinated context: ketchup is allowed
Reasoning: ketchup seems to be allowed…
Recommendation: use ketchup

Reasoning is ok (assuming context is true)

Relevancy

Question: can I add ketchup to hot dogs?
Fact: only mustard is allowed on hot dogs
Recommendation: add a lot of mustard

Looks like a valid answer to a different question

Precision

Question: can I add ketchup to hot dogs?
Fact: only mustard on hot dogs
Context: the grill temperature standard says..
Recommendation: no, only mustard

Answer is ok, but context is irrelevant to answer

Results

Usage stats

Company resources are important

- A few hundred tickets processed
- On average 45.1k tokens per ticket (95.2% read, 4.8% completions)
- Time to process tickets by agent: AVG: 25.1 seconds, MED: 22.3, 99%: 50.1s.
- CI workflow: 5 mins approx.
- About 0.12 USD per ticket in LLM costs at gpt-4o pricing per iteration.

Possible Improvements

Nothing is perfect

- The **summary creation part is not constrained enough** to represent total information of summary vs full standard. Human raters center around precision of the retrieved standards.
- Human feedback **does not show issues with Recall**, but more guardrails could be added here as well.
- In this version we **assumed the internal standards were totally consistent**, we are now creating tooling to know whether there are contradictions or gaps in their coverage.

Q & A

Thank you!