Securing Amazon Bedrock AgentCore

A Practical Framework

Rowan Udell • AWS Summit Sydney • April 2026

Agents Are Software.

Secure them like software.

What Makes Agents Different?

Traditional App

- Deterministic execution
- Fixed control flow
- Scoped permissions
- Predictable I/O

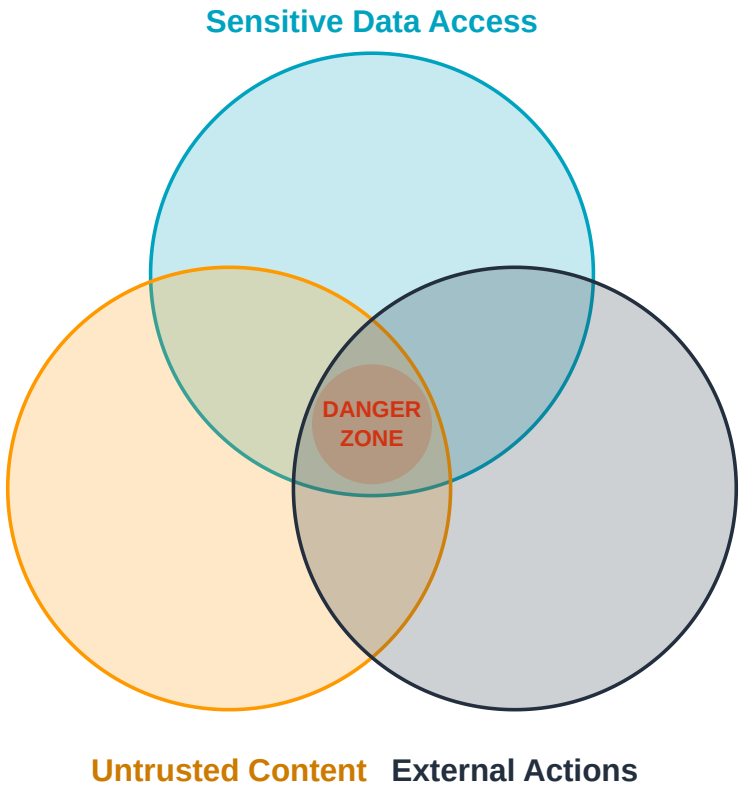
AI Agent

- Probabilistic decisions
- Dynamic tool selection
- Broad access patterns
- Untrusted content in the loop

The Lethal Trifecta

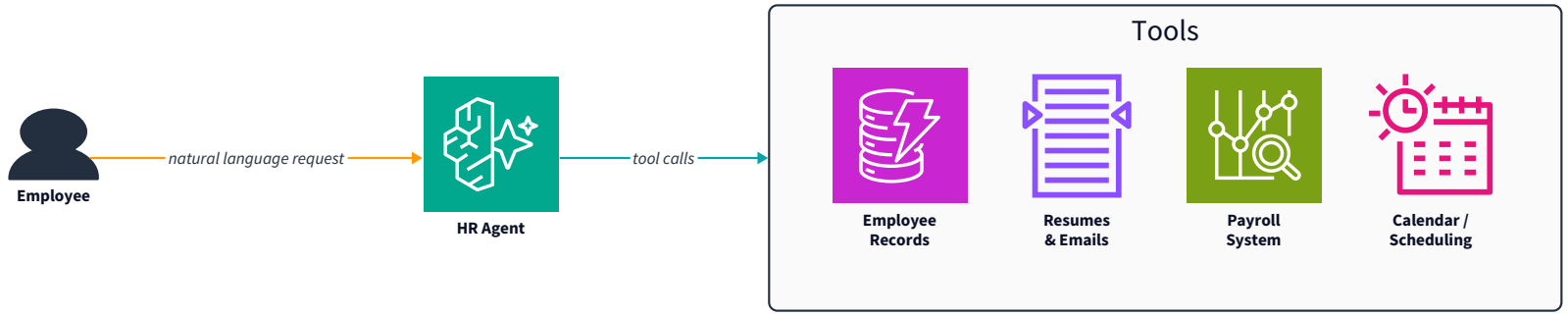
Simon Willison, 2025

Three Capabilities. Dangerous When Combined.



- Any two? Manageable.
- All three?
    i. **Data exfiltration.**
    ii. **Prompt injection exploitation.**
    iii. **Unauthorized actions.**

Meet the HR Assistant



An AI agent that helps employees with leave, payroll, onboarding

It has access to **employee records**, processes **messages** from users, and **takes actions** on other systems

Any concerns?

Sensitive Data Access

The agent can see what employees can't see about each other

- Employee records - names, salaries, performance reviews
- Payroll data - bank accounts, tax file numbers
- Leave balances and medical certificates
- Onboarding documents - ID copies, contracts

Untrusted Content

LLMs follow instructions in content, regardless of source

- Employee chat messages
- Uploaded resumes and CVs
- Forwarded emails
- RAG results from the knowledge base

External Actions

Data exfiltration as a feature

- Update payroll
- Send offer letters
- Book onboarding meetings
- Call external APIs

Guardrails that are 95% effective

are not reliable enough.

"Old school" security

is still your best friend.

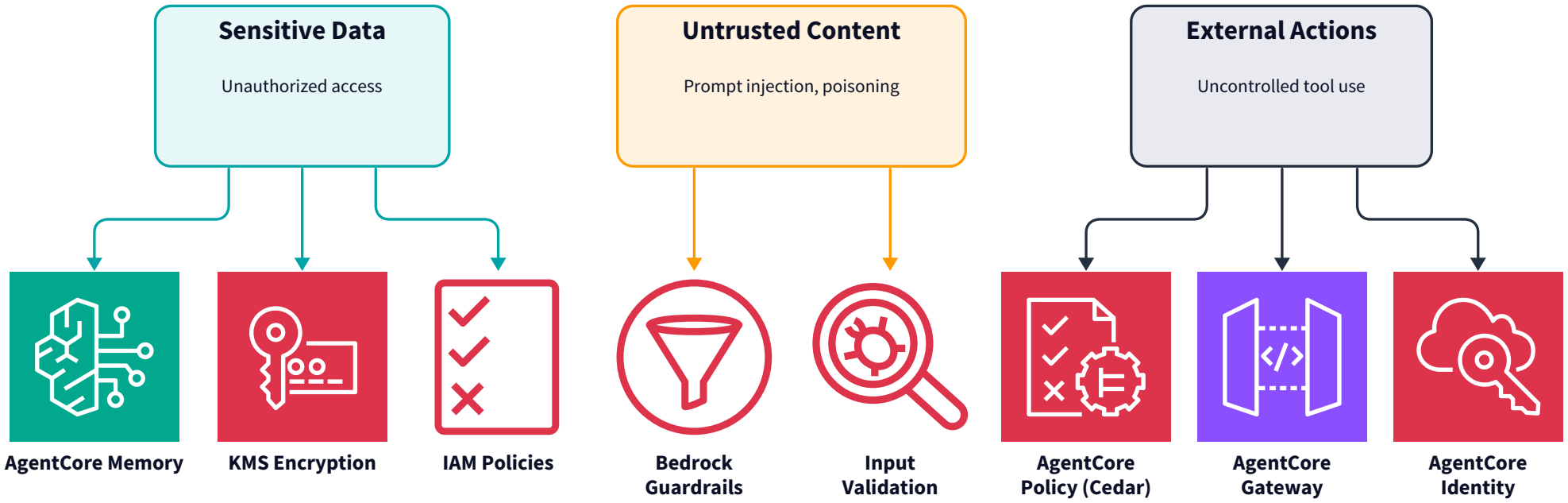
The Fundamentals Haven't Changed

- **Least privilege** - don't give agents permissions they don't need
- **Defense in depth** - multiple layers, not one guardrail
- **Separation of concerns** - isolate agent capabilities
- **Audit everything** - you can't secure what you can't see
- **Encrypt by default** - data at rest and in transit

AgentCore's Security Toolkit

Mapped to the Trifecta

The Map


Sensitive Data
Unauthorized access
Untrusted Content
Prompt injection, poisoning
External Actions
Uncontrolled tool use
AgentCore Memory
KMS Encryption
IAM Policies
Bedrock Guardrails
Input Validation
AgentCore Policy (Cedar)
AgentCore Gateway
AgentCore Identity

Leg 1: Controlling Data Access

AgentCore Memory

- Encryption at rest (KMS / CMK)
- Memory poisoning prevention
- Input validation

IAM Policies

- Least privilege per agent
- Resource-based policies on Runtime, Gateway, Memory

```
{
  "Effect": "Allow",
  "Action": [
    "bedrock-agentcore:GetMemory",
    "bedrock-agentcore:RetrieveMemoryRecords"
  ],
  "Resource":
    "arn:aws:bedrock-agentcore:
     ap-southeast-2:123456789012:
     memory/hr-assistant/*"
}
```

Not `bedrock-agentcore:*`. Never `*`.

Leg 2: Defending Against Untrusted Content

Bedrock Guardrails

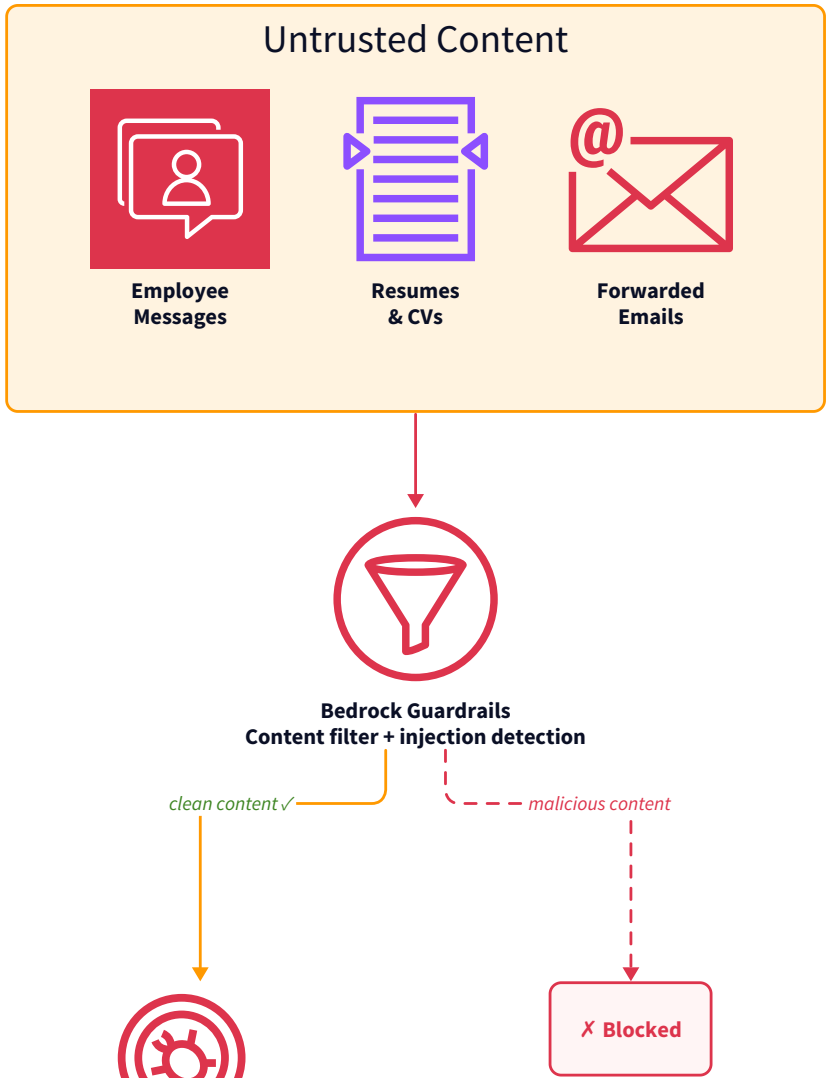
- Content filtering
- Prompt injection detection

Memory Hygiene

- Validate before storing
- Test for injection regularly

Prompt Engineering

- System prompts that resist manipulation



Leg 3: Controlling Tool Access

AgentCore Policy - Cedar policies, enforced **outside** agent code

```
// HR assistant can read employee records
permit(
  principal is AgentCore::OAuthUser,
  action == AgentCore::Action::"HRTools__get_employee_record",
  resource == AgentCore::Gateway::"arn:aws:bedrock-agentcore:ap-southeast-2:123456789012:gateway/hr-assistant"
) when {
  principal.hasTag("role") &&
  (principal.getTag("role") == "hr-manager" || principal.getTag("role") == "hr-admin")
};

// Nobody can bulk-export salary data
forbid(
  principal is AgentCore::OAuthUser,
  action == AgentCore::Action::"HRTools__export_salary_report",
  resource == AgentCore::Gateway::"arn:aws:bedrock-agentcore:ap-southeast-2:123456789012:gateway/hr-assistant"
);
```

The agent doesn't decide its own permissions. You do.

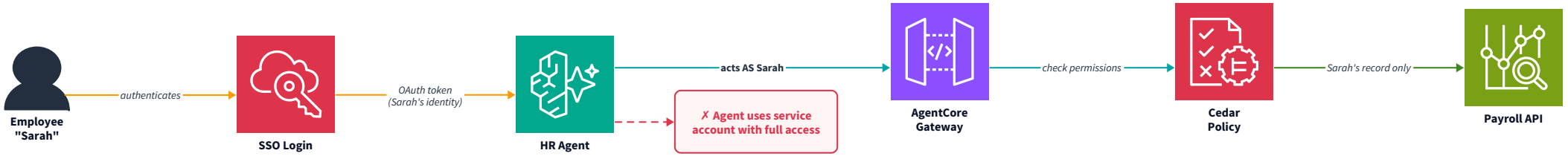
Leg 3: Gateway + Identity

AgentCore Gateway

- Centralized tool access
- Interceptors at 4 levels:
  - Gateway
  - Tool
  - Operation
  - Parameter

AgentCore Identity

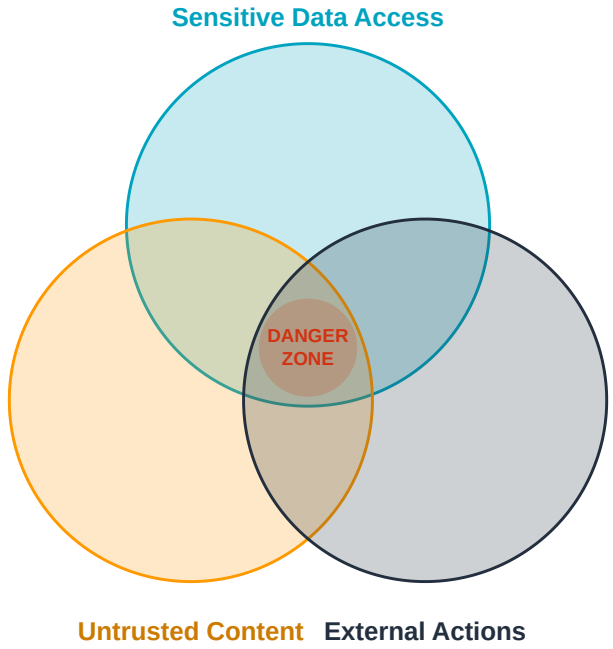
- OAuth 2.0 credential management
- Token vault
- Identity-aware authorization
- Agent acts **as** the user, not **instead of**



What To Do Tomorrow

Audit Your Agents

- Apply least privilege to agent IAM roles
- Use AgentCore Policy (Cedar) for tool boundaries - not agent code
- Encrypt memory, validate inputs, test for injection
- Enable CloudTrail + CloudWatch for agent activity
- Ask: does your agent **really** need all three legs?



Agents are software.

Secure them like software.

Thank You!

I help teams move agents from prototype to production 👋

linkedin.com/in/rowanu