CISCO Engage | Tech Day

Automation, AI and Segmentation

Sandeep Ramakrishnan
Delivery Architect

December 11 2025

Agenda

1. Objectives of this session
2. Automation with AI
3. Practical Examples
4. Securing AI

Why

- Been at Cisco 15 years
  - Been doing network security for twenty years
  - I am not a sw engineer
  - Adapt to the shift to devops model in customer environments
- Using AI for ZeroTrust
  - Practical tips and guidance
- Using ZeroTrust for AI
  - Security frameworks and controls
- Hopefully we all learn something new

Automation and AI

Engagement

What AI Does Well	Where AI Needs You
Generating content drafts quickly in multiple formats	Setting the purpose, audience, tone; refining for nuance, accuracy and impact
Processing and summarizing massive amounts of information in seconds	Guiding the focus to key points with added context and relevant details
Recommending diverse and potentially novel ideas from wide-ranging sources	Selecting the best ideas; adapting to strategy and feasibility
Detecting patterns and correlations across multi-source datasets	Validating findings; applying subject matter expertise to interpret meaning & application
Engaging in back-and-forth conversations and thought partnership	Asking, adjusting, and shaping until the output achieves desired goal

Automation Options in Segmentation

- Network Access Control Configuration
- Network Device Management
- Authentication Server Priority
- Account Creation
- Interface Configuration
- Validation of Deployment and Consistency

Automation Landing Points

Aspect	CLI Automation	API Automation
Interaction Method	Command-line commands (SSH)	Programmatic interfaces (RESTCONF, NETCONF, gNMI, SDKs)
Reliability	Rigid, syntax-dependent	Robust, model-driven, standardized
Maintenance	High, due to parsing and syntax issues	Lower, with reusable libraries and SDKs
Scalability	Limited, requires customization	High, supports complex workflows
Change Validation	Manual or limited scripting	Automated pre/post validation
Integration	Limited	Easy integration with DevOps and orchestration tools
Cisco Examples	Legacy expect scripts, CLI MOPs	Cisco NSO APIs, Cisco I

How to approach this

Scenario – Modify AAA server IPs on entire fleet

How to approach this

Step 1 - Plan

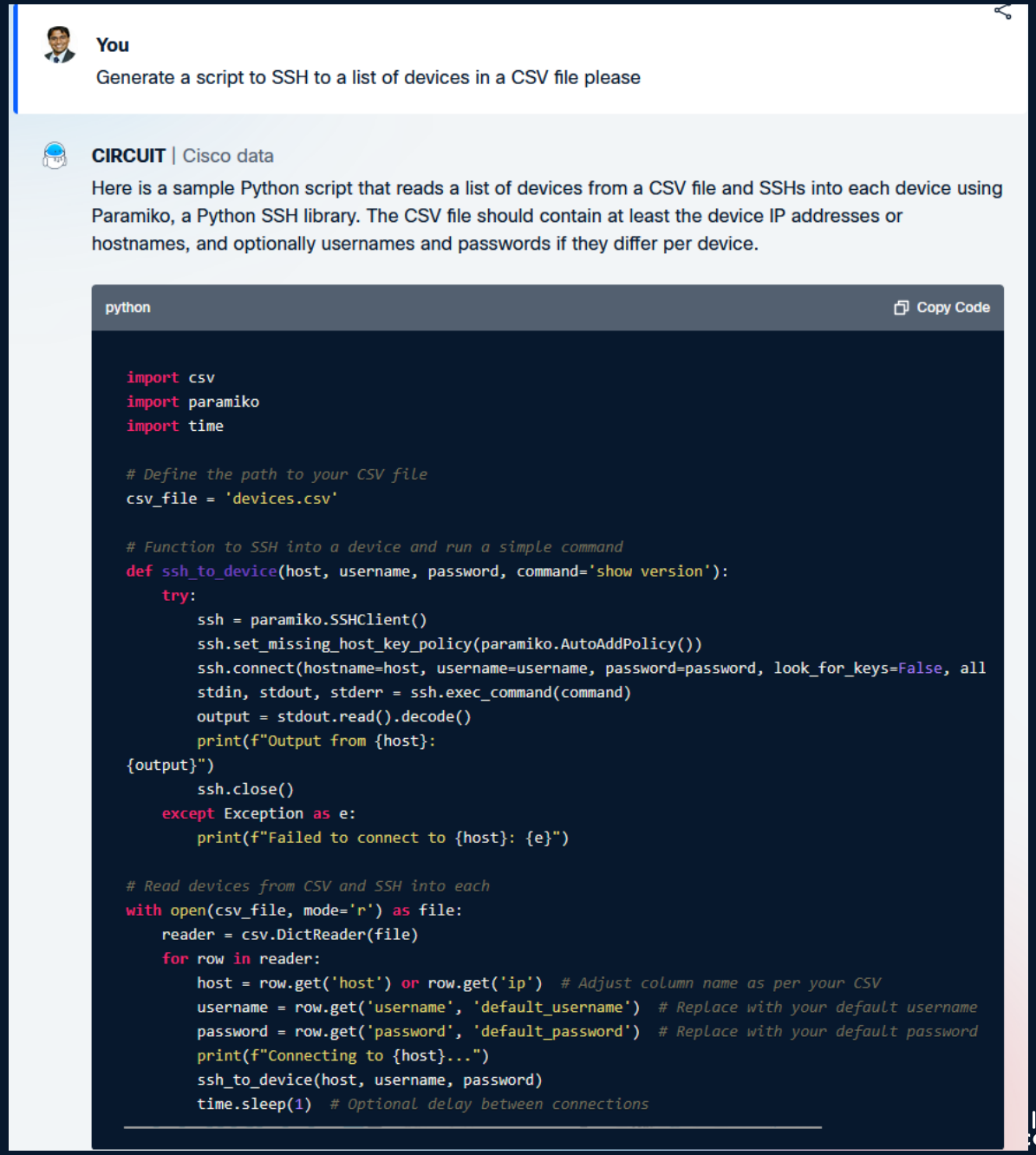
- Determine mode – API or CLI
  - Do the devices have management server with API capabilities
  - How many management servers
  - How many platforms
- Conceptualize the solution
  - Connect to each device
  - Update the AAA server
  - Validate that the change is working

- In this scenario – lets assume CLI because there are no management servers.
- So the solution will need to:
  - SSH to a list of device IPs
  - Run some pre-change checks
  - Execute commands to update AAA server
  - Run some post-change checks
  - Compare pre and post change

How to approach this

Step 2 – Build / Develop

- AI is very helpful here
- For each task, ask AI to help
  - Validate against known devices and tweak as necessary!



You

Generate a script to SSH to a list of devices in a CSV file please

CIRCUIT | Cisco data

Here is a sample Python script that reads a list of devices from a CSV file and SSHs into each device using Paramiko, a Python SSH library. The CSV file should contain at least the device IP addresses or hostnames, and optionally usernames and passwords if they differ per device.

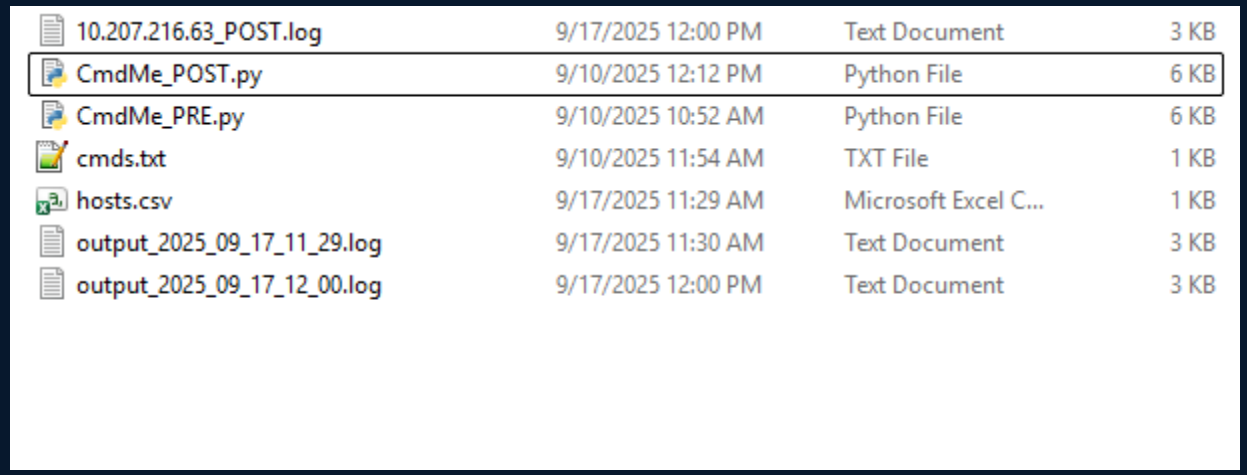
python — Copy Code

```
import csv
import paramiko
import time

# Define the path to your CSV file
csv_file = 'devices.csv'

# Function to SSH into a device and run a simple command
def ssh_to_device(host, username, password, command='show version'):
    try:
        ssh = paramiko.SSHClient()
        ssh.set_missing_host_key_policy(paramiko.AutoAddPolicy())
        ssh.connect(hostname=host, username=username, password=password, look_for_keys=False, all
        stdin, stdout, stderr = ssh.exec_command(command)
        output = stdout.read().decode()
        print(f"Output from {host}:
{output}")
        ssh.close()
    except Exception as e:
        print(f"Failed to connect to {host}: {e}")

# Read devices from CSV and SSH into each
with open(csv_file, mode='r') as file:
    reader = csv.DictReader(file)
    for row in reader:
        host = row.get('host') or row.get('ip')  # Adjust column name as per your CSV
        username = row.get('username', 'default_username')  # Replace with your default username
        password = row.get('password', 'default_password')  # Replace with your default password
        print(f"Connecting to {host}...")
        ssh_to_device(host, username, password)
        time.sleep(1)  # Optional delay between connections
```

How to approach this

Step 3 – Integrate and Optimize

- After building each task and testing
  - Get ready to merge and optimize
  - For my use case, this took me 4 hours to get a fully functional script



10.207.216.63_POST.log	9/17/2025 12:00 PM	Text Document	3 KB
CmdMe_POST.py	9/10/2025 12:12 PM	Python File	6 KB
CmdMe_PRE.py	9/10/2025 10:52 AM	Python File	6 KB
cmds.txt	9/10/2025 11:54 AM	TXT File	1 KB
hosts.csv	9/17/2025 11:29 AM	Microsoft Excel C...	1 KB
output_2025_09_17_11_29.log	9/17/2025 11:30 AM	Text Document	3 KB
output_2025_09_17_12_00.log	9/17/2025 12:00 PM	Text Document	3 KB

How would I approach this

Scenario – Automate Guest Password Generation

How to approach this

Step 1 - Plan

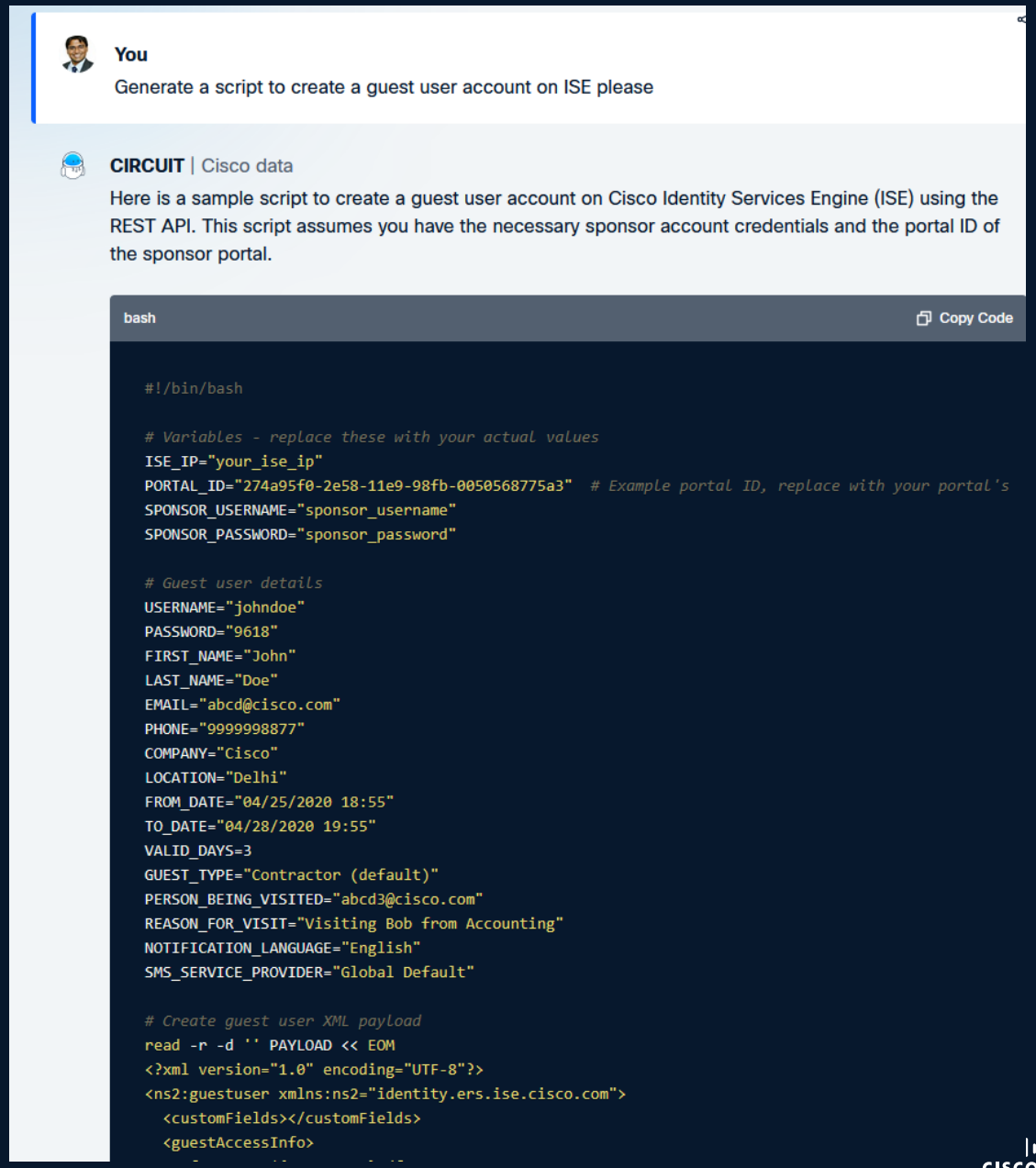
- Determine mode – API or CLI
  - Do the devices have management server with API capabilities
  - How many management servers
  - How many platforms
- Conceptualize the solution
  - Create A Guest User
  - Obtain the Password
  - Communicate the Account and Password to the Gues

- In this scenario, API since the Authentiation Server supports that
- So the solution will need to:
  - HTTPS to the AAA server
  - Create a Guest Account
  - Email or SMS that account to the end user
    - Possibly print it too!

How to approach this

Step 2 – Build / Develop

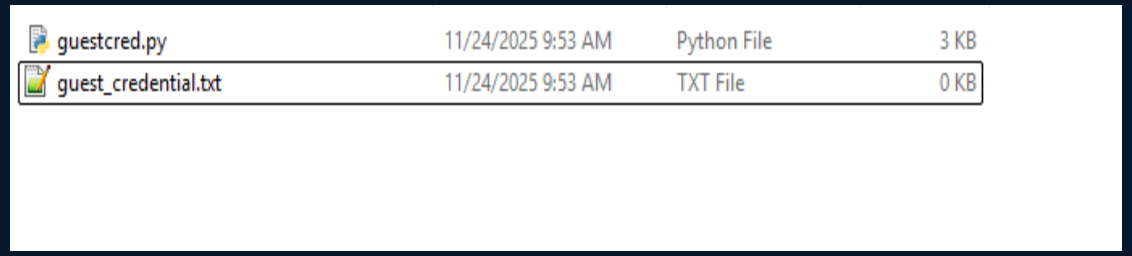
- AI is very helpful here
- For each task, ask AI to help
  - Validate against known devices and tweak as necessary!



How to approach this

Step 3 – Integrate and Optimize

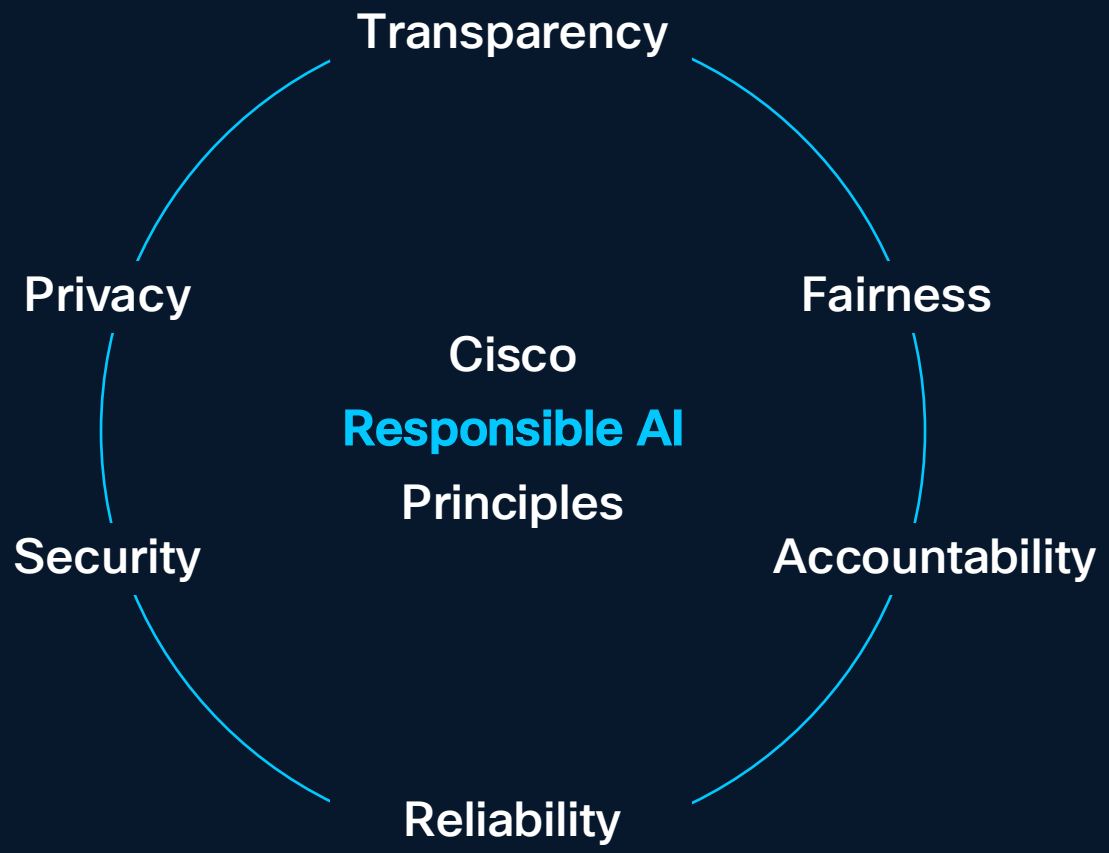
- After building each task and testing
  - Get ready to merge and optimize
  - For my use case, this took me 2.5 hours to get a fully functional script



Securing AI using Zero Trust Principles

Cisco AI Best Practices

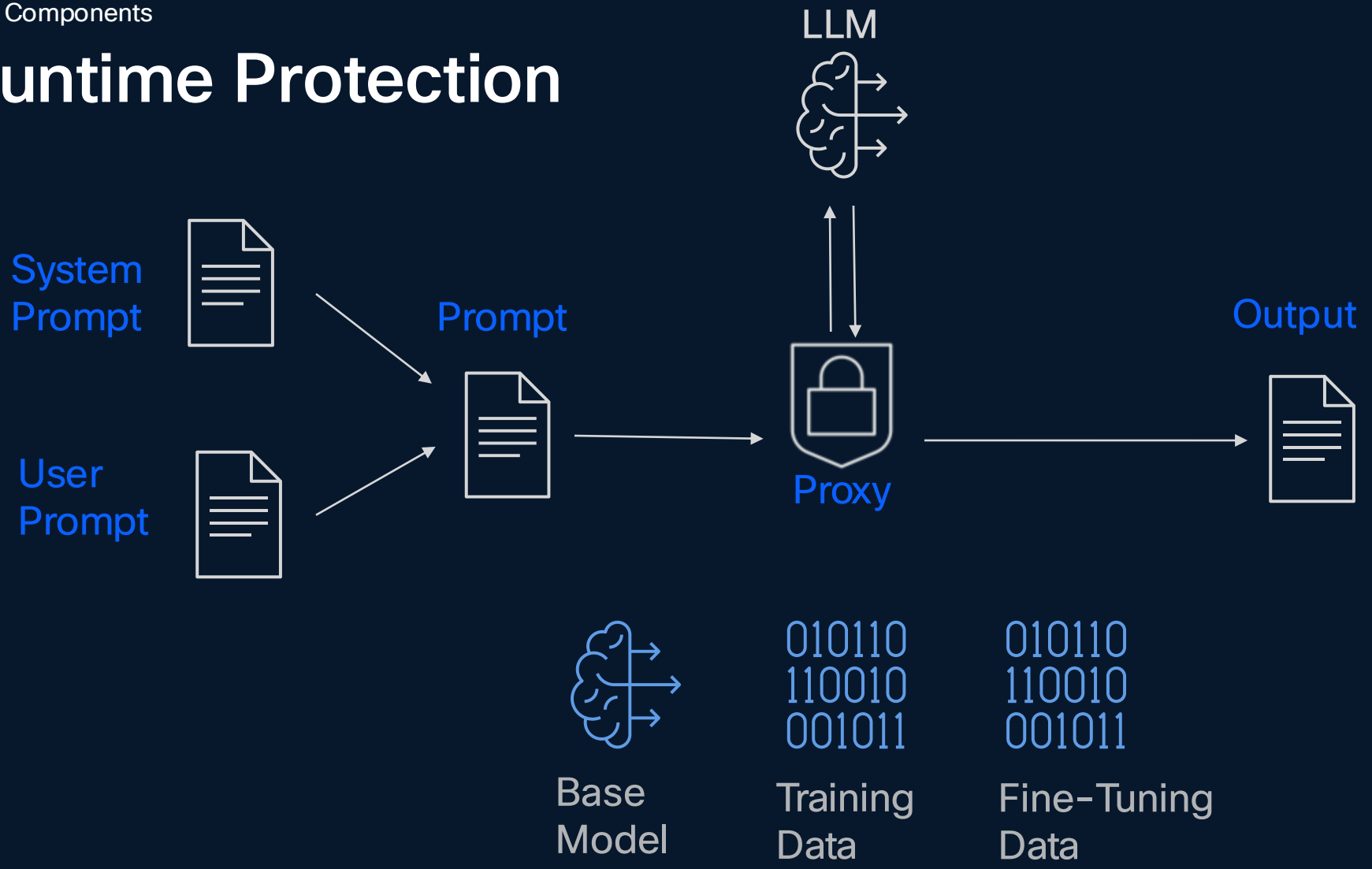
Preserving your trust with AI Governance



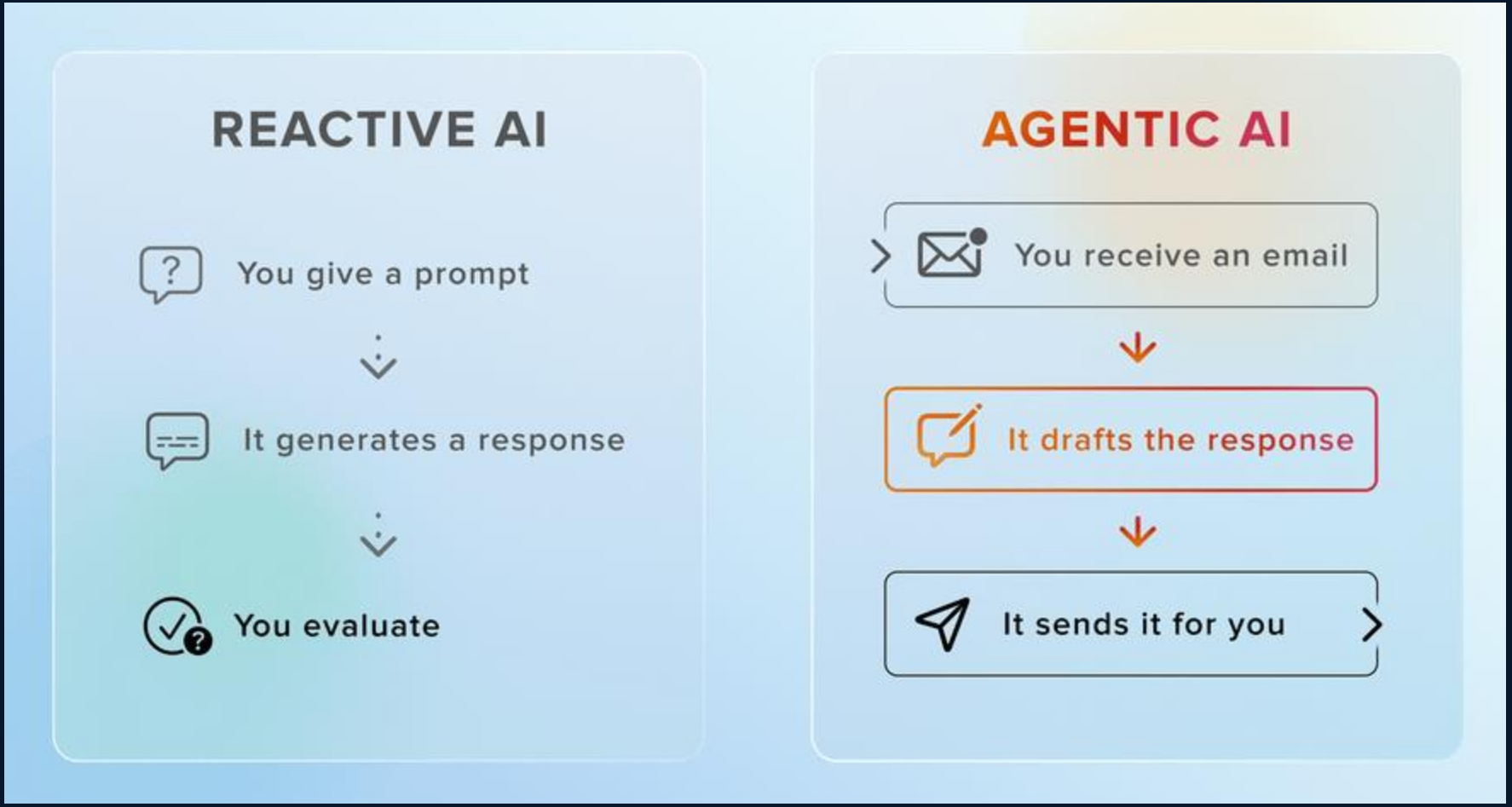
We're navigating the intersection of AI security, regulatory compliance, and Zero Trust—grounded in NIST 800-207, the NIST AI RMF, the EU AI Act, and informed by OWASP AI best practices, MITRE ATLAS threat models, and the Cloud Security Alliance (CSA) AI Controls Matrix.

AI Defense Components

AI Runtime Protection



Reactive and Agentic AI



Zero Trust Guardrails for AI Agents

Principle 1 – "Allow known good. Block everything else."

In a Zero Trust architecture, outlined in **NIST 800-207**, access is never implicit—it must be continuously validated. This principle is reinforced by **MITRE ATLAS**, which highlights how adversaries exploit ungoverned or over-permissive AI environments to deploy shadow agents or prompt injections.

OWASP's AI Guidelines recommend strict agent control mechanisms and validated allow lists to prevent rogue model execution. Likewise, the **EU AI Act (Articles 9–15)** mandates that high-risk AI systems operate within tightly controlled, pre-approved boundaries.

The **CSA AI Controls Matrix** emphasizes the importance of defining and enforcing strict access controls for AI systems, ensuring only authorized agents operate within predefined parameters.

Allow listing defines the safe zone. Everything else? It's denied by design.

Zero Trust Guardrails for AI Agents

Principle 2 – Make Policies Readable "Security that can't be understood, can't be trusted."

Declarative, transparent policies empower human oversight—vital per the **Govern function of the NIST AI RMF** and **Article 13 of the EU AI Act**, which require explainability in AI governance.

OWASP guidance emphasizes the importance of human-in-the-loop designs and interpretable rule enforcement. Security controls should be intelligible by policy owners, auditors, and compliance teams—not just engineers.

The **CSA AI Controls Matrix** aligns with this by advocating for clear documentation and transparency in AI system policies, facilitating easier audits and compliance checks.

This also supports Zero Trust policy centralization as specified in **NIST 800-207**, making enforcement both visible and auditable across domains.

Zero Trust Guardrails for AI Agents

Principle 3 – Log Everything "Every interaction tells a story. Capture it."

Comprehensive telemetry is critical. The **NIST AI RMF (Map & Measure)** calls for full lifecycle visibility, while **MITRE ATLAS** threat use cases demonstrate how visibility gaps enable stealthy AI model manipulation and misuse.

OWASP AI recommendations stress logging inputs, decisions, and outputs—particularly for model inference and external API interactions. The **EU AI Act (Article 12)** reinforces this, requiring audit trails that verify the integrity of decision-making processes.

The **CSA AI Controls Matrix** underscores the necessity of detailed logging and monitoring to detect anomalies and ensure accountability in AI operations.

Zero Trust requires pervasive observability, and that starts with capturing every event—authorized or blocked. If we can't understand the logs then that's a problem.

Zero Trust Guardrails for AI Agents

Principle 4 – Fail Closed, Not Open "No access is better than wrong access."

Failing open creates an adversary playground. **MITRE ATLAS** includes tactics where attackers induce fail-open conditions, such as DoS on policy evaluators or corrupt fallback mechanisms.

The **NIST AI RMF** emphasizes robustness and resilience—AI must respond to uncertainty by reducing exposure, not increasing it. The **EU AI Act (Article 14)** mandates fallback safeguards to ensure safety when anomalies are detected.

OWASP guidance advises strict policy enforcement with well-tested error handling to avoid insecure defaults.

The **CSA AI Controls Matrix** advocates for default-deny strategies and robust error handling to prevent unauthorized access during system failures.

In Zero Trust, indecision means denial—anything less is a liability.

Zero Trust Guardrails for AI Agents

Principle 5 – Use Multiple Layers "Defense in depth applies to AI too."

Layered controls are foundational in **NIST 800-207's ZTA model** and reinforced by the **Manage function of NIST AI RMF**. From input sanitization to model governance to contextual access checks—defense in depth is critical.

OWASP urges use of runtime monitors, rate limiters, and anomaly detectors. **MITRE ATLAS** documents chained attacks that bypass a single weak control—multi-layer defenses mitigate this.

The **CSA AI Controls Matrix** supports implementing multiple layers of security controls, including data encryption, access management, and continuous monitoring, to protect AI systems comprehensively.

Regulators agree: The **EU AI Act** calls for multi-dimensional risk controls across the AI lifecycle. Stack identity, environment, behavior, and purpose validation—because no single control is infallible.

Safe and Trustworthy Use of GenAI: Key Recommendations

Securing AI using Zero Trust Principals

- Compliance Strategy for Global Operations – Navigate data localization regulations and transfer mechanisms across geographies
- Embrace Privacy Regulation – Leverage growing public awareness of privacy laws and invest in compliance to build customer trust and mitigate reticence risk
- Privacy as a Business Enabler – Driving of public trust, agility, innovation, speed to market and operational efficiency
- AI Governance and Controls – Deploy AI responsibly by respecting privacy, managing unintended externalities, balance opportunities and risks
- Sustained Investment in Privacy and Security – Expect budgets to shift toward AI while ensuring AI investments support and strengthen privacy and security foundations

Source: April 22, 2025 Cisco's 2025 Data Privacy Benchmark Study sheds light on the importance of keeping sensitive information safe as AI transforms industries.


CISCO