Christopher Aaron O'Hara

Udacity MSc Capstone – Agentic AI Systems

February 21, 2026

Overview

This project implements a multi-agent cybersecurity workflow for local-cloud IIoT/OT incident triage and RCA support. The system uses one orchestrator and five specialized agents (intake, evidence, RCA, response planning, governance), integrates optional LLM refinement through a Vocareum-compatible OpenAI endpoint, and enforces policy constraints before final packet release. The objective is not autonomous remediation, but transparent, auditable decision support with explicit safeguards (Yao et al., 2023; National Institute of Standards and Technology, 2024).

Dataset: MITRE ATT&CK Enterprise STIX Enrichment

Source links: https://github.com/mitre/cti/tree/master/enterprise-attack ; https://www.nist.gov/cyberframework

Project repository: https://github.com/Ohara124c41/agentic-ai-mitre-attack-rca.git

Motivation

Industrial SOC environments require fast triage under uncertainty, yet unconstrained assistant outputs are difficult to verify and govern. The scenario dataset used in this project contains heterogeneous conditions (risk scores 0.56-0.91, evidence counts 1-3, mixed anomaly tags, and a prompt-injection case), which makes it suitable for testing bounded agentic behavior rather than one-pass generation. A staged agentic design was selected so each decision step can be inspected, logged, and validated against policy and zone-conduit constraints (Stouffer et al., 2023; ISA, n.d.).

Task and Use Case Description

The task is semi-autonomous incident reasoning with constrained action recommendation. The system receives an incident scenario, extracts risk/evidence context, proposes RCA hypotheses, maps ATT&CK techniques, drafts an action plan, and applies governance checks before producing a final decision packet. This is a single orchestrated workflow with five role agents and tool interactions, where outputs are expected to be observable and auditable rather than hidden in a monolithic response (MITRE, n.d.; Yao et al., 2023).

In the provided run, the system processed 5 incidents and produced:

- 4 `escalate` outcomes and 1 `refuse` outcome,
- `policy_pass_rate = 0.4`,
- `mean_evidence_quality = 0.74`,
- `mean_hypothesis_confidence = 0.8187`,
- live model activity (`llm_calls_count = 5`).

System Architecture and Workflow Design

The architecture is defined as one orchestrator with explicit handoffs to five role agents:

1. Intake Agent
2. Evidence Agent
3. RCA Agent
4. Response Planner Agent
5. Governance Agent

The orchestrator maintains shared incident state and writes auditable outputs (`decision_packets.jsonl`, `audit_log.jsonl`, state snapshots). A structured architecture description is provided in Figure 1 below.

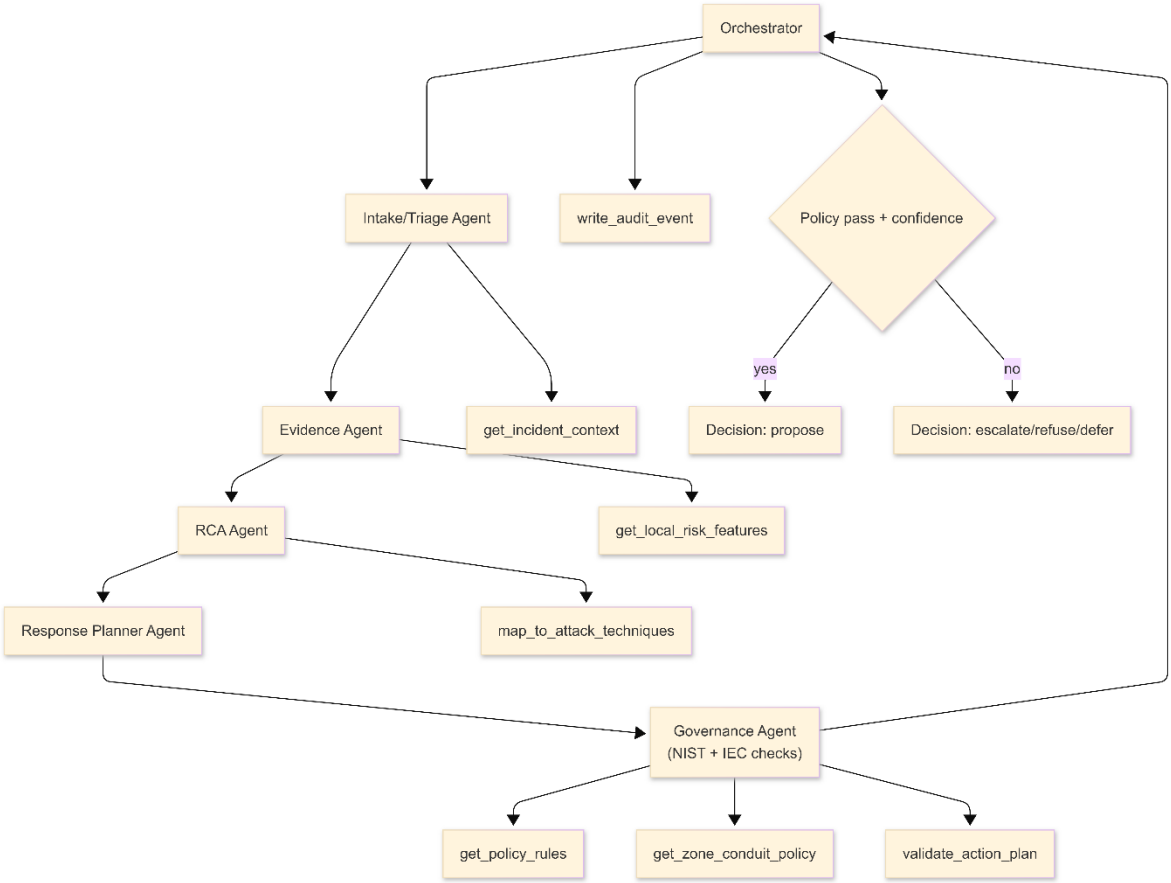

Figure 1 - High-level system architecture of agentic system.

Design justification: scenario heterogeneity and policy-sensitive actions make explicit sequencing necessary. A single assistant response cannot expose which step produced a conclusion, while staged handoffs enable stage-level replay and latency profiling.

Persona, Reasoning, and Decision Logic

The system uses role-specific reasoning:

- Intake infers severity from risk features and flags prompt-injection markers.

- Evidence computes evidence quality from bundle support.

- RCA builds competing hypotheses, assigns confidence, and maps ATT&CK techniques.

- Response planning drafts bounded operational actions.

- Governance enforces action-policy and zone-policy constraints and determines escalation requirement.

Observed behavior in this run confirms deterministic decision logic plus model-assisted RCA text refinement:

- prompt-injection scenario (`INC_C_PROMPT_INJECT`) was refused,

- all policy-failing cases were escalated,

- stage ordering was consistent across incidents (`intake -> evidence -> rca -> response -> governance`).

Tool Use and Memory Design

Tool integrations include:

- scenario/context loading tools (`incident_tools.py`),

- ATT&CK mapping tools (`attack_mapping.py`),

- governance/policy tools (`governance_tools.py`),

- audit logging (`audit_tools.py`),

- optional OpenAI-compatible LLM tool for hypothesis refinement (`llm_client.py`).

Memory/state is handled through a typed incident state object (`IncidentState`) that stores:

- scenario context,

- evidence bundle and quality score,

- hypothesis candidates and selected hypothesis,

- mapped techniques and proposed actions,

- policy findings, logs, and trace IDs.

This design supports replayable reasoning rather than stateless chat output.

Safety, Reliability, and Transparency

Safety controls implemented:

- prompt-injection marker detection in intake,

- policy allow-list and action-cap checks,

- IEC zone-conduit action constraints,

- confidence/evidence gating before policy pass,

- escalation path enforcement,

- refusal path for unsafe instruction patterns.

Transparency controls implemented:

- JSONL audit events with stage/event metadata,

- per-incident state snapshots,

- LLM call traces with prompt hashes and response previews,

- runtime summary with explicit `llm_runtime` diagnostics.

These controls convert runtime behavior into inspectable evidence rather than implicit logic.

Evaluation of Agent Behavior

Runtime behavior metrics

From `outputs/run_summary.json`:

- `run_count = 5`

- `decision_types = {propose: 0, escalate: 4, refuse: 1, defer: 0}`
- `escalation_rate = 1.0`
- `policy_pass_rate = 0.4`
- `mean_evidence_quality = 0.74`
- `mean_hypothesis_confidence = 0.8187`
- `llm_runtime.llm_calls_count = 5`

Handoff and latency

Handoff replay shows complete stage coverage for all incidents. Latency concentration occurs in the RCA stage (median stage delta about 2.08 seconds), while other stage transitions are near-zero in this environment. This is consistent with optional LLM call overhead being concentrated in RCA.

ATT&CK and RCA output quality

The run produced ATT&CK mappings across six techniques: `T1021, T1048, T1071, T1078, T1562, T1569`.

Weighted multi-agent consensus benchmarking yielded:

- consensus label accuracy: `0.8`
- expected-technique hit rate: `1.0`

Per-agent benchmark accuracy:

- intake: `0.8`
- evidence: `0.8`
- RCA: `0.6`
- governance: `0.6`
- response: `0.4`

Interpretation: technique-level coverage is strong, while label-level disambiguation remains a realistic improvement target (MITRE, n.d.).

Governance V&V

Custom V&V checks in notebook all passed in this run:

- required NIST refs present,

- IEC zone-conduit check present,

- prompt-injection not auto-proposed,

- policy failures imply escalation,

- action plans bounded (`<= 3` actions).

Experimental Comparison and Design Justification

Although the primary artifact is one agentic workflow, the notebook includes an operational comparison axis:

- deterministic fallback vs LLM-enabled RCA enrichment,

- side-by-side hypothesis output comparison,

- incident-linked LLM call trace verification.

This comparison is meaningful because it isolates the contribution of model-assisted wording while preserving identical orchestration and policy logic. In the current run, connectivity and traces confirm active LLM usage, but action/governance outcomes remain controlled by explicit rules rather than model freeform output.

Baseline vs. Experimental Comparison

Comparison design:

- Baseline configuration: deterministic RCA enrichment (`use_llm = False`), executed on the same 5 incidents.

- Experimental configuration: LLM-enabled RCA enrichment (`use_llm = True`, `gpt-4o-mini`) on the same incidents.

- Held constant: scenario set, orchestrator, agent order, policy rules, governance thresholds, and output schema.

Side-by-side outcomes:

- Decision distribution: baseline `escalate=4, refuse=1` vs experimental `escalate=4, refuse=1` (no change).

- Policy pass rate: baseline `0.4` vs experimental `0.4` (no change).

- Mean hypothesis confidence: baseline `0.8187` vs experimental `0.8187` (no change).

- LLM call count: baseline `0` vs experimental `5` (expected change).

- RCA phrasing diversity:

  baseline `unique_response_ratio=0.2`, `mean_pairwise_similarity=1.0`;

  experimental `unique_response_ratio=1.0`, `mean_pairwise_similarity=0.5770`.

Interpretation:

The controlled comparison shows that enabling LLM enrichment changed RCA wording diversity while leaving governance-relevant outcomes unchanged. This matches the implemented design: confidence scoring and policy checks are deterministic, while LLM is limited to text refinement.

Interpretation for a Non-Technical Audience

This system acts like a structured analyst assistant. It reads an incident case, checks evidence, proposes likely causes, and recommends cautious next steps, but only after policy and safety checks. In this run, the system frequently chose escalation and once refused an unsafe instruction-style case, which indicates it is designed to prioritize safety over automation speed. The tool can help analysts organize investigations faster, but final operational decisions still require human review.

Ethical and Responsible Use Considerations

Three project-specific concerns are evident:

1. Over-automation risk: high escalation rates can reduce missed detections but may increase analyst workload and alert fatigue.

2. Uneven control impact across zones: OT scenarios had higher policy-violation incidence in this run, which can bias where strict controls are triggered.

3. Instruction-channel misuse: prompt-injection style content can influence model text if safeguards are weak; refusal and policy gating are required protections.

Operational fairness was screened with AIF360-compatible logic using a non-degenerate adverse outcome (`policy_violation_flag`) (Bellamy et al., 2019; Hardt et al., 2016):

- OT adverse rate: `0.75`

- IT adverse rate: `0.00` (small IT sample)

These values are screening signals, not deployment-ready fairness conclusions, due to sample size.

Observed Behavior and Limitations

Observed strengths:

- complete and auditable handoff chain,

- enforced escalation on policy failure,

- robust refusal behavior for prompt-injection scenario,

- high ATT&CK technique hit rate in scenario benchmark.

Observed limitations:

- small sample (`n=5`) limits statistical stability,

- decision distribution is conservative (no `propose` outcomes),

- LLM RCA phrasing shows partial homogenization across scenarios,

- fairness metrics are high-variance with minimal IT support.

Limitations, Risks, and Safeguards

Key risks and mitigations:

- Risk: policy over-triggering in OT contexts.

  Safeguard: explicit zone-policy constraints and logged violation reasons.

- Risk: opaque LLM influence.

  Safeguard: incident-linked LLM trace logging and deterministic fallback support.

- Risk: false confidence from small benchmark.

  Safeguard: explicit small-sample caveat and replayable artifacts for iterative validation.

Future Improvements

1. Expand scenario set to reduce metric variance and improve fairness interpretability.

2. Add calibrated confidence thresholds to separate `propose` from `escalate` more effectively.

3. Improve RCA diversity with stronger retrieval grounding and anti-homogenization prompts.

4. Add richer policy packs (IEC zone rules, change-window controls, asset criticality priors).

5. Add uncertainty-aware scoring to consensus RCA benchmark.

Future Work: Integrative Industry Synthesis

This agentic AI system project is designed to integrate with earlier capstone stages as an orchestration/governance layer. In integrated industrial applications project, it can consume upstream model outputs

(statistical, ML, deep, generative) and enforce policy-bound action routing with traceability. The intended value is system-level assurance: transparent handoffs, explicit constraints, and auditable final decisions aligned with industrial security governance.

Reproducibility

Reproducibility controls included:

- script-first runtime (`agentic_runtime.py`),

- notebook `Run All` workflow with deterministic checks,

- `.env` credential pattern via `utils.py`,

- minimal project-scoped `requirements.txt`,

- persisted outputs (packets, logs, snapshots, embeddings, model snapshot),

- separate baseline comparison outputs (`outputs_det`) for deterministic-vs-LLM analysis.

The full workflow can be replayed from notebook or runtime script with identical scenario inputs and configuration.

Technology, Method, and Concept Citation Coverage

Technologies used and cited:

- Python, Jupyter, NumPy, Pandas, OpenAI API-compatible interface, Matplotlib, Seaborn, NetworkX, AIF360.

- Runtime/dependency support libraries used in this project path: Requests, python-dotenv, SciPy, scikit-learn, Fairlearn-compatible fairness stack components, and notebook kernel runtime support via IPython/ipykernel (Bird et al., 2020; Granger et al., 2021).

Methods and concepts used and cited:

- agent orchestration patterns and tool-use reasoning,

- staged decision logic and bounded action planning,

- ATT&CK-based technique mapping,

- governance checks aligned with NIST CSF and IEC zone-conduit principles,

- parity-based fairness-risk screening with small-sample caveats.

References

1. Bellamy, R. K. E., Dey, K., Hind, M., Hoffman, S. C., Houde, S., Kannan, K., Lohia, P., Martino, J., Mehta, S., Mojsilovic, A., Nagar, S., Ramamurthy, K. N., Richards, J., Saha, D., Sattigeri, P., Singh, M., Varshney, K. R., & Zhang, Y. (2019). AI Fairness 360: An extensible toolkit for detecting and mitigating algorithmic bias. IBM Journal of Research and Development, 63(4/5), 4:1-4:15. https://doi.org/10.1147/JRD.2019.2942287
2. Bird, S., Dudik, M., Edgar, R., Horn, B., Lutz, R., Milan, V., Sameki, M., Wallach, H., & Walker, K. (2020). Fairlearn: A toolkit for assessing and improving fairness in AI. Microsoft Research. https://fairlearn.org/
3. Hardt, M., Price, E., & Srebro, N. (2016). Equality of opportunity in supervised learning. In Advances in Neural Information Processing Systems 29. https://proceedings.neurips.cc/paper/2016/hash/9d2682367c3935defcb1f9e247a97c0d-Abstract.html
4. Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., Fernandez del Rio, J., Wiebe, M., Peterson, P., ... Oliphant, T. E. (2020). Array programming with NumPy. Nature, 585, 357-362. https://doi.org/10.1038/s41586-020-2649-2
5. Hagberg, A. A., Schult, D. A., & Swart, P. J. (2008). Exploring network structure, dynamics, and function using NetworkX. In Proceedings of the 7th Python in Science Conference (pp. 11-15). https://networkx.org/documentation/stable/reference/introduction.html
6. Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. Computing in Science & Engineering, 9(3), 90-95. https://doi.org/10.1109/MCSE.2007.55
7. ISA. (n.d.). ISA/IEC 62443 series of standards. https://www.isa.org/standards-and-publications/isa-standards/isa-iec-62443-series-of-standards

8. McKinney, W. (2010). Data structures for statistical computing in Python. In S. van der Walt & J. Millman (Eds.), Proceedings of the 9th Python in Science Conference (pp. 56-61). https://doi.org/10.25080/Majora-92bf1922-00a
9. MITRE. (n.d.). ATT&CK STIX data (Enterprise). GitHub. https://github.com/mitre/cti/tree/master/enterprise-attack
10. National Institute of Standards and Technology. (2024). The NIST Cybersecurity Framework (CSF) 2.0. https://www.nist.gov/cyberframework
11. OpenAI. (n.d.). API reference. https://platform.openai.com/docs/api-reference
12. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825-2830. https://jmlr.org/papers/v12/pedregosa11a.html
13. python-dotenv. (n.d.). python-dotenv documentation. https://saurabh-kumar.com/python-dotenv/
14. Project Jupyter. (n.d.). Project Jupyter documentation. https://docs.jupyter.org/en/latest/
15. Reitz, K. (n.d.). Requests: HTTP for humans. https://requests.readthedocs.io/en/latest/
16. Granger, B. E., Perez, F., & Jupyter Development Team. (2021). IPython and Jupyter kernel architecture documentation. https://ipython.readthedocs.io/
17. Stouffer, K. A., Pease, M., Tang, C., Zimmerman, T., Pillitteri, V. Y., Lightman, S., Hahn, A., Saravia, S., Sherule, A., & Thompson, M. (2023). Guide to Operational Technology (OT) Security (NIST SP 800-82r3). National Institute of Standards and Technology. https://doi.org/10.6028/NIST.SP.800-82r3
18. Van Rossum, G., & Drake, F. L. (2009). Python 3 reference manual. CreateSpace. https://docs.python.org/3/
19. Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., ... SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. Nature Methods, 17, 261-272. https://doi.org/10.1038/s41592-019-0686-2
20. Waskom, M. L. (2021). seaborn: Statistical data visualization. Journal of Open Source Software, 6(60), 3021. https://doi.org/10.21105/joss.03021
21. Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., & Cao, Y. (2023). ReAct: Synergizing reasoning and acting in language models. In International Conference on Learning Representations (ICLR). https://arxiv.org/abs/2210.03629