COALITION FOR SECURE AI

AI Incident Response Framework, V1.0

Workstream 2: Preparing Defenders for a Changing Cybersecurity Landscape

Version 24fecc8c5c0b102456eb748cdae00f18f70bdc7a

27 October 2025

Contents

AI Incident Response Framework, V1.0

Approved by the CoSAI Project Governing Board on 27 October 2026

OASIS Open Project : Coalition for Secure AI (CoSAI) - Workstream 2: Preparing Defenders for a Changing Cybersecurity Landscape

1. Overview

1.1. Abstract

This paper addresses the topic of incident response in the context of AI systems. As with other materials produced by the Coalition for Secure AI, this paper focuses on technological capabilities and gaps that are specific to the field of artificial intelligence and does not address the topic of incident response in other contexts as this has been well-researched and documented by existing frameworks (such as NIST Incident Response Recommendations and Considerations for Cybersecurity Risk Management). This paper guides defenders on proactively minimizing the impact of AI system exploitation. It details how to maintain auditability, resiliency, and rapid recovery even when a system is compromised by advanced threat actors. The guide also explores the unique challenges of AI incident response, emphasizing the role of forensic investigation and the complications introduced by agentic architectures, while providing concrete steps to manage this new complexity.

1.2. How To Use This Document

1.2.1. Quick Start Guide

The incident response life cycle has several versions from varying sources. For the purposes of this paper we will focus on NIST 800-61r3's life cycle phases and provide a mapping to the closest SANS PICERL phases as a reference.

NIST SP 800-61 rev 3 lifecycle phase	Closest SANS PICERL phase(s)	AI relevance
Govern (GV)	Preparation	Set AI-specific policy and roles: e.g., who may push new model weights to production, required update cadence for model cards, and breach-disclosure windows for LLM supply-chain compromises.
Identify (ID)	Preparation	Maintain inventories of datasets, model artefacts, prompts, and dependencies; perform AI-threat risk assessments (model-exfiltration, data-poisoning, jailbreaks) and maintain model card library.
Protect (PR)	Preparation	Apply safeguards: encrypt checkpoints, enforce least-privilege on inference APIs, implement input-validation/output-filtering guardrails, and add watermarking to deter prompt-injection abuse.
Detect (DE)	Identification	Ensure proper Monitoring and Telemetry

NIST SP 800-61 rev 3 lifecycle phase	Closest SANS PICERL phase(s)	AI relevance
Respond (RS)	Containment & Eradication	Contain (hot-swap to safe snapshots, throttle APIs, revoke leaked keys), eradicate root causes, and coordinate disclosure to affected users.
Recover (RC)	Recovery	Recreate or retrain from trusted baselines, run robustness/bias test suites, and validate checkpoints against performance and fairness thresholds before full go-live.
Identify.IM – Improvement	Lessons Learned	Feed incident artefacts into future Reinforcement Learning from Human Feedback (RLHF) or adversarial-training cycles and relevent montior tools, SIEMs, etc., update threat models and playbooks, share anonymised TTPs

1.2.2. Reader Guidance

This framework is designed for a diverse audience, from technical teams to executive leaders. Not every reader will need every section in equal depth. We outline which sections of the document each audience should prioritize, followed by recommendations on how to use the framework in practice.

Note: If you are new to this document, you may start with the Quick Start Guide (Section 1.2.1) for a brief overview of the AI incident response lifecycle. This guidance will help you navigate efficiently, whether you're skimming for a high-level overview or planning an in-depth implementation.

Audience	Introduction	Details	Reference / Standards
CISO / Security leader	1.3 Executive Summary	3.4 Roles & Responsibilities 3.5 Communication (incl. 3.5.1 Regulatory)	4.1 NIST SP 800-61r3 Alignment
Engineer / AI Developer	3.1 Pre-Incident Prep 3.2 Monitoring & Telemetry	3.3 IR Workflow	6 Technical Architecture Patterns
Incident Responder (SOC/CSIRT)	2 Understanding AI Incidents	3.3 IR Workflow (incl. 3.3.3.1 Forensics) 4.3 Playbooks	3.4 Roles & Responsibilities 3.5 Communication
Policymaker / Compliance	1.3 Executive Summary	3.1 Prep (policies, third-party) 3.5.1 Regulatory	4.1 NIST Alignment
Technical PM / Project Lead	1.3 Executive Summary 2 Incident Types	3.1 3.2 3.3 (what to build & run)	3.4 Who does what

1.3. Executive Summary

The increasing integration of Artificial Intelligence (AI) into business processes presents a new frontier of risks. AI systems, with their inherent complexity and autonomy, can have significant consequences, ranging from reputational damage and financial loss to legal and ethical liabilities. To navigate this evolving landscape, a specialized AI Incident Response Framework is a necessity. This document provides an executive summary of the AI Incident Response Framework developed by COSAI, designed to equip security teams with the necessary capabilities to effectively manage AI-specific incidents. This framework provides a structured approach to preparing for, detecting, responding to, and learning from AI incidents, ensuring organizational resilience in the age of AI.

1.3.1 Types of AI Incidents

A shared taxonomy sharpens detection, triage, ownership, and playbook design.

A. Performance & Robustness Failures

- **Model Drift:** Accuracy or calibration declines as data distribution shifts.
- **Adversarial Evasion:** Crafted inputs bypass filters or controls.
- **Unexpected/Harmful Outputs:** Nonsensical, policy-violating, or unsafe generations.

B. Security Vulnerabilities (ML Pipeline & Model)

- **Data Poisoning:** Training data manipulation alters model behavior.
- **Model Inversion / Membership Inference:** Sensitive training data leakage via queries.
- **Model Theft / Extraction:** Unauthorized replication of proprietary models.

C. Ethics, Fairness & Bias Incidents

- **Algorithmic Bias:** Systematic disparate impacts across protected classes.
- **Opacity / Lack of Explainability:** Decisions cannot be justified in regulated contexts.
- **Unintended Social Consequences:** Downstream harm to users or society.

D. Misuse & Abuse

- **Deepfakes/Disinformation:** Synthetic media for fraud or manipulation.
- **AI-Assisted Attacks:** AI leveraged to scale phishing, malware, and fraud.

1.3.2 AI Incident Response Plan

Adapts the traditional incident response lifecycle to the specific challenges of AI. It provides a structured, repeatable process to ensure timely and effective management of AI incidents. The plan is organized into five key phases:

1. Preparation

- Complete production model inventory (owners, data, risks, criticality).
- Define AI-IR roles, authorities, and decision thresholds.
- Maintain AI-specific playbooks (bias, drift, adversarial, data leakage).
- Run training & simulations (tabletops, red/blue prompt exercises).

2. Detection & Analysis

- Monitor performance, safety, fairness, and security signals (e.g., drift stats, toxicity, jailbreak patterns, data exfiltration indicators).
- Enable intake channels (internal reports, user complaints, vendor alerts).
- Triage by severity, blast radius, regulatory exposure, and patient/customer impact.

3. Containment

- Disable tools or rollback to prior model; gate high-risk features.
- Isolate systems; rate-limit or block malicious inputs.

- Activate temporary guardrails (stricter content filters, human-in-the-loop).

4. Eradication & Recovery
   - Root cause analysis (data, model, infra, policy).
   - Retrain/patch models; cleanse data; rotate keys/tokens.
   - Validate performance, safety, and compliance before redeploy.

5. Post-Incident Activities
   - Blameless post-mortem, timeline, and effectiveness review.
   - Action items with owners and deadlines; update playbooks, policies, and training.
   - Feed lessons into ML governance, risk registers, and CI/CD controls.

1.3.3 Key Roles & RACI

Clear accountability avoids delay and rework. Example matrix below; tailor to your org.

Activity	Incident Commander	AI/ML Engineer	Legal & Compliance	Communications
Declare Incident	A	R	I	I
Technical Investigation	C	A/R	I	I
Take Model Offline	A	R	C	I
External Communications	C	I	A	R
Post-Mortem Report	A	R	C	C

Legend: R = Responsible, A = Accountable, C = Consulted, I = Informed.

1.3.4 Incident Response Playbooks

Playbooks translate policy into actionable detections. Each should include the following structure:

- **Incident Type:** A clear description of the specific AI incident the playbook addresses.
- **Detection:** How to identify that this type of incident is occurring, including specific monitoring alerts and reporting channels.
- **Triage and Severity:** Criteria for assessing the severity of the incident and the key stakeholders to notify.
- **Containment Steps:** A checklist of immediate actions to take to contain the incident.
- **Eradication and Recovery Procedures:** Detailed technical steps for resolving the issue and restoring the system.
- **Communication Plan:** Pre-approved communication templates for internal and external stakeholders.

Some example playbooks:

- Adversarial Evasion (Prompt/Content Bypass)
- Model/Data Exfiltration or Leakage

1.3.5 Key Lessons Learned

- Run blameless post-mortems within SLA; capture timeline, root causes, and control gaps.
- Produce measurable actions (owners, deadlines) and track to closure.

- Instrument governance: add fairness/safety KPIs to production SLOs; expand telemetry (prompt logs, content filters, tooling traces); integrate security scanning for AI/ML in CI/CD.
- Prevent recurrences: clarify authority to disable/rollback; expand adversarial testing; standardize fallback modes and comms templates.

2. Understanding AI Security Incidents

AI security incidents can be categorized into five primary domains based on the attack vector and impact area:

2.1. Data Incidents

Incident Type	*Description*	*Example*
Data Poisoning	Deliberate contamination of training data to induce specific model behaviors	Injecting fake reviews into a dataset to make an AI model favor specific products artificially
Data Leakage	Unauthorized access to or exfiltration of sensitive data	Hackers gaining access to confidential medical records used to train an AI model
Data Integrity Violations	Tampering with AI system inputs to manipulate outcomes	Modifying sensor data in a self-driving car to make it misinterpret road signs

2.2. Model Incidents

Incident Type	*Description*	*Example*
Model Theft/Extraction	Unauthorized duplication or stealing of model weights and architecture	Reverse engineering a proprietary chatbot's model to replicate its behavior
Model Inversion	Extracting private training data from model responses	Using an AI model's outputs to reconstruct sensitive user data, like medical histories
Backdoor Attacks	Hidden functionality implanted in models that activates under specific conditions	Adding a secret trigger phrase to make an AI classify harmful content as safe
Model Fooling	Fooling models by slight modifications	Placing a tape over a 85 mph sign, causing an autopilot AI to read it as 35 mph

2.3. Deployment Incidents

Incident Type	*Description*	*Example*
Prompt Injection	Manipulating model inputs to bypass safeguards or alter system behavior	Sending a specially crafted prompt to make an AI assistant disclose restricted information
Indirect Prompt Injection	Manipulating external data that the model references to bypass safeguards or alter system behavior	Planting malicious content in locations that an AI uses to make an AI assistant disclose restricted information
Jailbreaking	Circumventing model safety measures to access restricted functionality	Using creative prompts to make a chatbot generate unethical or harmful content
Abuse Generation	Harmless prompts trigger harmful output	Benign prompts causing profane and inappropriate responses
Rogue Agentic AI	Agentic AI taking unexpected actions	Closing a curtain when the user says a harmless "thanks" in response to a chat message

2.4. Infrastructure Incidents

Incident Type	*Description*	*Example*
Unauthorized Access	Gaining privileged access to AI infrastructure	Exploiting a vulnerability to gain admin access to an AI-powered recommendation system
API/Service Abuse	Excessive or malicious use of AI services	Overloading an AI translation service with excessive requests to disrupt its functionality
Dependency Exploitation	Attacking AI dependency libraries	Exploiting vulnerabilities in the libraries an AI is built upon

2.5. User Interaction Incidents

Incident Type	*Description*	*Example*
Malicious Code Generation	Leveraging AI to produce functional malware	Using an AI to create polymorphic malware bypassing detection
Phishing Content Generation	Leveraging AI crafting high-quality, convincing phishing content	Cybercriminals using an AI to craft flawless phishing emails
Output Manipulation	Forcing harmful or misleading outputs from AI systems	Crafting inputs to make an AI generate false financial forecasts
Hallucination Exploitation	Leveraging model inaccuracies for harmful purposes	Using an AI's hallucinated data to spread misinformation in public forums
Regional or cultural bias	Not acceptable otput of the model in some regions or cultures	Exploiting AI's specifics to provide different outputs for different or cultures

3. Creating an AI Incident Response Plan

The **core objectives** of an AI Incident Response Plan are to: - Minimize business and user impact during incidents - Protect sensitive data from AI system exploitation - Maintain regulatory compliance - Preserve evidence for forensic analysis - Identify root causes for prevention and response - Restore secure AI system functionality - Share actionable intelligence with security community

The AI Incident Response Plan described is limited in scope to the: - AI systems covered (LLMs, RAG systems, agentic architectures; further discussed in Section 6) - Integration points with traditional IR procedures and frameworks (further discussed in Section 4)

3.1. Pre-Incident Preparation

Component	Key Elements
Risk Assessment	· Secure AI system components and architecture · Critical data assets processed by AI systems · Potential attack vectors based on ATLAS framework · Compliance requirements · Business impact of potential compromise
IR Team Composition	· AI/ML engineers with model expertise · Data scientists for pipeline understanding · Security analysts with AI knowledge · Legal and compliance experts · Business stakeholders · Communications specialists
Baseline Telemetry	· Input/output logging for all interactions · System prompt and configuration versioning · Tool usage tracking (agentic systems) · Authentication and authorization events · Embedding and vector database query patterns · Resource utilization metrics

3.2. Monitoring and Telemetry

To protect AI systems from evolving threats, telemetry must capture a comprehensive set of signals spanning model inference behavior, prompt and output risks, content safety, and model integrity. This includes monitoring for prompt injection, output manipulation, knowledge base poisoning, model drift, unauthorized tool or API usage, and other adversarial activities. Tracking agent workflows, context exchanges, and system lifecycles provides visibility into misuse, operational anomalies, and attacks. This data enables organizations to safeguard AI pipelines, ensure compliance, and strengthen incident detection and response with traceable, structured observability data.

Telemetry Type	Purpose	Relevant Threats Detected
Model Inference Activity	Tracks all inference executions including token usage, confidence, latency, and endpoints.	Abuse of model APIs, data exfiltration, inference abuse (e.g., model scraping).

Telemetry Type	Purpose	Relevant Threats Detected
Prompt Injection Detection	Monitors input prompts for adversarial payloads or jailbreak attempts.	Prompt injection attacks, jailbreaking, prompt leakage.
Content Risk Detection	Assesses LLM output for toxicity, bias, PII leakage, and unsafe content generation.	Disinformation, output poisoning, PII/PHI leakage, model misuse.
Model Drift Detection	Detects distributional shifts in model input/output indicative of adversarial manipulation or operational risk.	Data poisoning, concept drift, model degradation over time.
Agentic Workflow Execution	Logs multi-step decision-making by AI agents, including tool calls and retries.	Malicious tool chaining, unauthorized data access, misuse of plugins/APIs.
MCP (Model Context Protocol) Message Activity	Observes protocol-level context exchanges in agentic and retrieval-augmented systems.	Session hijacking, tampering with context/memory, unauthorized prompt modifications.
Agentic RAG Workflow Execution	Captures the steps of retrieval-augmented generation pipelines including retrievers, LLMs, and validators.	Knowledge base poisoning, unsafe content retrieval, response manipulation.
AI Incident Finding	Records confirmed or suspected incidents detected across AI systems (e.g., poisoning, unauthorized use, guardrail events).	Incident response triggers — prompt abuse, model misuse, security violations.
Tool Call Execution	Logs API/tool calls made by agents, including argument inspection and error rates.	Unauthorized external access, misuse of internal or third-party APIs.
AI System Activity	Lifecycle monitoring of AI systems: deployments, updates, shutdowns, failures.	Unauthorized model deployments, version tampering, system compromise.

Effective security monitoring of AI systems requires capturing fine-grained telemetry at each of the input, processing, and output stages of AI workflows. Below are the some of the attributes essential for building threat detection use cases, anomaly detection, and incident response for AI-driven environments:

Attribute	Explanation
timestamp	Tracks the time of activity performed. Proper recording of time enables investigators to piece a timeline together especially when AI incidents span multiple services or layers.
identity	Tracks the identity that performed an action. This could be a user or machine identity.
prompt_text	Captures the raw input text or structured request sent to the AI model. Monitoring the prompt_text is crucial for detecting prompt injection attacks, data exfiltration attempts, or adversarial instructions that could manipulate model behavior.
completion_text	Captures the model's output or response. It is essential to inspect completion_text for toxic content, bias, hallucinations, or leakage of sensitive information such as PII, credentials, or internal data.

Attribute	Explanation
model_name/version	Logs the name and version of the model used. This ensures that only authorized and validated models are deployed and helps detect unauthorized model swapping or shadow AI deployments.
token_usage	Measures the number of tokens in both input and output. Unusually high or low token_usage patterns can signal abusive scraping attempts or denial-of-service (DoS) behavior aimed at exhausting AI model resources.
latency_ms	Measures the inference response time in milliseconds. Sudden spikes or drops in latency_ms may indicate resource exhaustion attacks, model misuse, or service degradation due to adversarial workloads.
retrieved_docs	Tracks documents fetched during Retrieval-Augmented Generation (RAG) workflows. It helps detect knowledge base poisoning, unauthorized document access, or retrieval manipulation that could compromise model outputs.
tool_calls	Logs all external or internal tools invoked by AI agents. Monitoring tool_calls helps uncover unauthorized access to sensitive APIs, malicious tool chaining, or exfiltration via tool misuse during autonomous agent operations.
confidence_score	Provides the model's confidence in its prediction or generation. Very low confidence scores can signal model uncertainty or adversarial influence, while unusually high confidence on manipulated inputs could reveal model evasion attempts.
drift_metrics	Measures data drift or distribution changes in model inputs/outputs over time. Significant drift could indicate model poisoning, concept drift attacks, or environmental changes that degrade model performance or security.
trace_id/session_id	Correlates activities across sessions and distributed systems. Proper tracking of trace_id and session_id enables forensic investigations, cross-system correlation, and incident scoping when AI incidents span multiple services or layers.

3.3. Incident Response Workflow

3.3.1. Preparation Phase

Activity	AI-Specific Considerations	Implementation Guidance
Risk Assessment & Threat Modeling	· Identify critical assets in each architecture pattern · Map ATLAS threat vectors to components · Assess likelihood and impact · Prioritize security controls · Document risk thresholds	· Use architectural patterns from Section 6 · Reference ATLAS techniques in security planning · Conduct architecture-specific threat modeling · Prioritize controls for high-impact scenarios

Activity	AI-Specific Considerations	Implementation Guidance
Monitoring Infrastructure	· Input/output logging · System prompt versioning · Configuration management · Authentication tracking · API usage analysis · Resource utilization · Data access patterns	· Implement privacy controls for prompt logging · Maintain auditable prompt history · Track all model versions and parameters · Establish API usage baselines · Monitor for anomalous resource consumption
Response Capability Development	· Specialized AI security training · Response tools for AI systems · Emergency access procedures · Backup procedures · Recovery testing	· Train teams on AI-specific threats · Implement tools for log analysis and prompt filtering · Create secure backups of models and embeddings · Regularly test restoration procedures

3.3.2. Detection and Analysis Phase

Activity	AI-Specific Considerations	Key Considerations
Detection Mechanisms	**Automated Monitoring:** · LLM-based classifiers for suspicious interactions · Semantic similarity checks · Token usage anomaly detection · User feedback signal monitoring **Manual Review:** · Human review procedures for flagged interactions · Sampling of high-risk operations · Security dashboards · Escalation triggers **Integration Points:** · SIEM integration · Correlation with network security · Authentication event linking	· Balance automated detection with human oversight · Establish clear thresholds for alerts · Avoid false positives through baseline calibration · Ensure privacy considerations in monitoring · Correlate AI system events with broader security telemetry · Implement strong AI security controls for all LLM-based monitoring solutions

Activity	AI-Specific Considerations	Key Considerations
Initial Triage	**Incident Verification:** · Confirm security incident status · Classify per AI attack categories (Section 2) · Determine affected components **Impact Assessment:** · Estimate compromise scope · Identify affected users/data · Assess data exposure issues · Determine business impact **Priority Assignment:** · Assign priority based on impact · Implement notification procedures · Mobilize specialized resources	· Leverage AI to aggregate data needed for initial assessment of incident · Validate whether anomalous behavior constitutes a security incident · Use categories from Section 2 for classification · Focus on potential data exposure through AI systems · Consider regulatory implications · Ensure escalation procedures match incident severity
Investigation Procedures	**Evidence Collection:** · Interaction logs · System configurations · Model versions and parameters · Network traffic and API calls · Vector database queries **Forensic Analysis:** · Malicious prompt patterns · Data retrieval sequences · Tool usage analysis · Memory manipulation assessment · Data poisoning evaluation **Attribution Assessment:** · Targeted vs. opportunistic analysis · Actor identification · Threat intelligence correlation · Attribution documentation	· Use AI to assist in analyzing and summarizing evidence documents · Preserve evidence in a forensically sound manner · Focus on the unique aspects of AI system compromise · Analyze both the explicit content and implicit patterns · Consider the non-deterministic nature of AI systems · Document investigation methodology thoroughly

3.3.3. Containment, Eradication, and Recovery Phase

Activity	AI-Specific Considerations	Implementation Guidance
Containment Strategies	**Short-term Containment:** · Emergency prompt filters · Rate limiting/access restrictions · Component isolation · User/IP blocking **Architecture-Specific Containment: • Basic LLM: **Input validation, reduced temperature, content filtering** • LLM with Memory: **Reset memory, session isolation, conversation history purging** • RAG: **Data source quarantine, validation of retrieval results** • Agentic: **Disable tools, enhance authorization, least privilege checks** • Agentic RAG:** Combined strategies with interface security focus **Evidence Preservation:** · Forensic copies before containment · Documented containment actions · Vector database state preservation · Model weight copies	· Implement containment without destroying evidence · Consider each architectural pattern's unique vulnerabilities · Focus containment on the specific attack vector · Document all containment actions thoroughly · Maintain chain of custody for forensic evidence · Consider business impact of containment measures
Eradication Procedures	**Root Cause Elimination:** · Vulnerability addressing · Enhanced input validation · System prompt updates · Framework patches **Component Remediation: • Data-Level: **Remove poisoned data, reconstruct embeddings** • Model-Level: **Roll back to secure versions, enhance guardrails** • Deployment-Level: **Update prompts, strengthen controls** • Output-Level:** Implement filtering, content moderation **Verification Testing:** · Penetration testing · Attack reproduction attempts · Red-team exercises · Issue verification	· Address fundamental vulnerabilities, not just symptoms · Apply remediations specific to the compromised level · Test thoroughly before returning to production · Document all changes for future reference · Maintain audit trail of remediation actions · Verify effectiveness through testing

Activity	AI-Specific Considerations	Implementation Guidance
Recovery Procedures	Service Restoration: · Capability prioritization · Phased approach planning · Verification criteria · Required approvals **Enhanced Monitoring:** · Heightened monitoring deployment · Normal/abnormal metrics · Additional logging · Scheduled reviews **User Communication:** · Communication templates · Transparency guidelines · Communication protocols · User feedback mechanisms **Business Continuity:** · Alternative system activation · Restoration prioritization · Stakeholder expectation setting · Impact documentation	· Implement a gradual, verified restoration process · Maintain heightened monitoring during recovery · Communicate clearly with affected users · Document business impact throughout recovery · Ensure leadership approval at key recovery stages · Balance security with operational needs

3.3.3.1. Forensics for AI Systems

In order to effectively conduct forensic analysis of security incidents on AI systems, the AI system should be able to record (and provide) sufficient amount of information for the investigator. For example, it is essential to be able to retrieve both system prompt and user prompts and any other parameters which were used when interacting with an AI model as well as the response from the system. In general AI model will not produce exactly the same outputs even given the same input and system prompt, therefore it is essential to preserve raw output details from the AI model in order to be able to reconstruct the whole picture of the incident.

AI systems can also interface and integrate with external functions and components (such as MCP servers). As with traditional security, accurate logging of these interactions and a log trail of external function executions should be preserved and made available to the forensic investigator.

User inputs may be used as a part of future training data for an AI system, therefore it is essential to be able to track user input inclusion and impact on training datasets.

AI-based tools can also be used for effective analysis of forensic trails after an incident. However, similar to how traditional forensic tools can be attacked and exploited by malicious attacker - the investigator should be aware that AI-enabled analysis and forensic tools can also become a target of malicious activity. The risks of AI-based tool analysis should be evaluated based upon your organizations risk tolerances and mitigations should be leveraged including the use of sandboxed, isolated environments.

Clearly, in this section we describe the expectations when there are no constraints. However, in real world AI system implementation resources can be finite and system constraints may impact the scope and scale of data preservation. As in traditional systems, the best practice is to identify data retention objectives at design time and to address resource constraints via data preservation techniques and data rotation during the implementation and deployment of AI system.

During analysts the investigator can ask the following questions: * Were any of the guardrails successfully bypassed? * Were any of the prompts intentionally modified by the attacker? * Did the system produce unexpected and potentially harmful content? * Did content include any sensitive information that could be potentially exfiltrated by an attacker? * Did any other manipulation of data or system take place?

Finally, it is essential to understand and investigate the external components that were interracting with AI systems. Log trails of these components (such as web server logs, database query logs and so on) could also be extremely helpful in being able to reconstruct and understand the full picture of an AI security incident.

3.3.4. Post-Incident Activity Phase

Activity	Key Components	Implementation Guidance
Lessons Learned	**Post-Incident Review:** · Timely meeting scheduling · Cross-team representation · Event timeline documentation · Response effectiveness analysis **Root Cause Analysis:** · Underlying vulnerability investigation · Determination of causation: · Architectural design flaws · Security control implementation errors · Operational failures · Novel attack techniques **Response Evaluation:** · Detection effectiveness assessment · Containment strategy evaluation · Recovery efficiency analysis · Business impact measurement	· Schedule reviews within 1-2 weeks of resolution · Focus on systemic improvements, not blame · Document timeline from detection through resolution · Identify gaps in tools, processes, and training · Analyze whether incident resulted from known or novel techniques · Measure response metrics against defined objectives

Activity	Key Components	Implementation Guidance
Security Enhancement	**Architectural Improvements:** · System architecture updates · Additional security layers · Integration point strengthening · Component separation enhancement **Control Enhancements:** · New/improved security controls · Prompt engineering updates · Monitoring capability enhancement · Access control strengthening **Process Refinement:** · Playbook updates · Role/responsibility revision · Communication protocol improvement · Documentation enhancement	· Implement architectural changes to address root causes · Strengthen controls at the affected system level · Update detection mechanisms for similar threats · Revise processes based on response effectiveness · Document all improvements for future reference · Assign clear ownership for enhancement implementation
Knowledge Sharing	**Internal Transfer:** · Case study documentation · Lesson-sharing workshops · Security awareness updates · Executive briefings **External Sharing:** · Industry knowledge contribution · Information-sharing participation · MITRE ATLAS submissions · Standards organization engagement **Regulatory Reporting:** · Required notifications · Compliance documentation · Security measure evidence · Audit record maintenance	· Leverage AI in drafting incident reports · Create training materials from incident findings · Share lessons across security and AI teams · Contribute to industry knowledge responsibly · Complete all regulatory reporting requirements · Balance transparency with security considerations · Document compliance with relevant regulations

3.4. Roles and Responsibilities

3.4.1. Provider vs. Consumer Responsibilities

A shared responsibility model between providers and consumers is essential to establish clear accountability and separation of duties in order to safeguard AI systems effectively. Providers—including base model developers, agent developers, and API hosts—should be responsible for ensuring foundational model robustness, implementing safety alignment, securing inference infrastructure, and embedding standardized guardrails. Consumers—comprising application developers and end organizations—must manage prompt and context control, application-level filtering, compliance monitoring, and incident detection.

3.4.2. Team Assessment and Training

Activity	Description	Implementation Guidance
Tabletop Exercises	Scenario-based discussions using AI attack cases from Section 2	· Conduct quarterly exercises · Rotate scenario types · Include cross-functional teams · Document lessons learned
Technical Drills	Hands-on exercises focused on specific AI incident response tasks	· Practice log analysis · Test containment procedures · Validate recovery processes · Measure response times
Cross-Functional Exercises	Full-team simulations involving all relevant stakeholders	· Include business representatives · Test communications channels · Validate escalation procedures · Practice regulatory notifications
Adversarial Simulations	Red-team exercises based on ATLAS techniques	· Simulate prompt injection attacks · Test data poisoning scenarios · Practice response to model extraction · Evaluate detection capabilities
Skills Assessment	Regular evaluation of team knowledge and capabilities	· Identify training gaps · Update learning materials · Track improvement metrics · Recognize high performers

3.5. Communication and Collaboration

3.5.1. Regulatory Communication

Incident reporting has different requirements for different jurisdictions and industry verticals, manifold across levels of government. In the US the 2022 CIRCIA legislation provided strict timelines for reporting cyber incidents to critical infrastructure segments even tighter than the financial industry's GLBA for financial privacy requirements, while HIPAA requires report of PHI breaches over longer timescales. All 50 US states have their own breach notification laws. In contrast, the EU is governed by NIS2 for critical infrastructure, DORA for financial institutions and GDPR for personal data breaches - all on strict timelines. Legal counsel is needed to fully understand the reporting requirements across jurisdictions.

AI Systems bring new challenges to reporting for regulatory compliance: * Personal data breaches from PII/PHI leakage from AI systems, applicable to US states' data laws and GDPR * Model confidentiality, integrity and accessibility attacks are breaches, even if affected indirectly * AI service providers may be considered critical infrastructure, or be required to disclose incidents affecting customer use of AI services

3.5.2. Information Sharing with Community

One of defenders' greatest advantages against attackers is information sharing, as timely well-structured information about attacks enables defenders to prevent similar attacks. Incident information sharing is evolving from sector based ISAC reporting to more strict national and supranational reporting requirements from CIRCIA, CISA in the US and ENISA in the EU. There are many fragmented and boutique reporting mechanisms too, including the AI Incident Database, for community-driven reporting. Cross-border open groups such as FIRST offer best practices and standards for sharing IOCs, TTPs and for Coordinated Vulnerability Disclosure. The security community supports the use of OASIS STIX 2.1 to share information widely, with OpenCTI the intended target open source tool for managing threat intelligence within or across organizations. Open sharing of threat intelligence is still a challenge for enterprises and even the security community, often because complying with data privacy regulation discourages additional anonymization work required.

AI incident reporting will often require: * Data scrubbing to remove PII from incident report, possibly data used by AI system * Associating IOCs and TTPs with attacks against the AI system * Responsible disclosure of AI vulnerabilities to AI providers

4. Frameworks and Playbooks

4.1. NIST SP 800-61r3 Alignment

Reference: NIST SP 800-61r3

The NIST SP 800-61 Revision 3 (2025) modernizes incident response by embedding it within the broader context of cybersecurity risk management through alignment with the NIST Cybersecurity Framework (CSF) 2.0. Unlike its predecessor, which focused on procedural guidance, this version adopts a flexible, outcome-driven approach using the six CSF Functions—Govern, Identify, Protect, Detect, Respond, and Recover—to guide organizations in managing, mitigating, and learning from cybersecurity incidents. The framework emphasizes continuous improvement, cross-functional coordination, and real-time learning across incident stages, integrating threat intelligence, asset visibility, and policy enforcement into an adaptive lifecycle. For AI systems and generative AI applications, the framework is especially applicable as it supports incident response for AI-specific threats such as prompt injection, model inversion, hallucinations, data leakage, adversarial misuse of toolchains (e.g., LangChain agents), and retrieval augmentation abuse. It enables organizations to adapt incident handling playbooks, define AI-relevant response roles, log and monitor AI inputs/outputs, and apply continuous learning to secure model and agent behaviors and limit data exposures. Through its flexible and modular structure, SP 800-61r3 provides a resilient foundation for defending dynamic, intelligent, and increasingly autonomous AI ecosystems.

4.1.1. Key Concept

SP 800-61r3 shifts from static procedural guidance to a flexible, CSF 2.0-aligned risk management profile, suitable for dynamic, evolving technologies like AI systems.

It replaces the old linear IR lifecycle with a model mapped to the six CSF 2.0 Functions: Govern, Identify, Protect, Detect, Respond, and Recover — all of which are critical for generative AI environments where risks emerge through prompt injection, model behavior, data flow, and tool orchestration.

CSF Function	AI-Specific Considerations
Govern (GV)	Establish AI-specific IR policies: include prompt injection, misuse of RAG pipelines, and training data exposures. Ensure third-party responsibilities (e.g., model vendors or SaaS orchestration) are contractually defined.
Identify (ID)	Inventory LLM tools, APIs, model endpoints, external connectors and data sources. Capture attack surfaces such as embeddings, memory modules, toolchains (LangChain, ReAct). Threat modeling must include jailbreaking and adversarial prompts.
Protect (PR)	Implement robust input/output guardrails, system prompts, and behavioral constraints. Ensure LLM telemetry, prompt logs, and PII filters are enabled. Validate training pipelines to prevent data poisoning or unintentional PII ingestion.
Detect (DE)	Continuously monitor prompt patterns, output anomalies, and system behavior. Correlate logs from LLM APIs, tool executions, and chat sessions. Integrate LLM-as-a-Judge metrics and fine-tuned filters to detect AI-targeted abuse.
Respond (RS)	Coordinate across AI/ML teams, SREs, legal, and security to investigate and contain AI-specific incidents like data leakage or unauthorized tool invocation. Update response plans to reflect LLM threats and recovery protocols.
Recover (RC)	Restore compromised AI services, rollback fine-tuned models or agent states, redact unsafe memory or embeddings. Update RAG sources and retrain if training data was involved in the breach. Conduct post-mortems with AI-specific lens.

4.1.2. Key Recommendations

NIST Recommendation	*AI System Implication*
Establish IR policy & procedures	Include AI-specific incident types and response playbooks for AI misuse, such as unauthorized API use, hallucinations, or data leakage.
Communication protocols	Define rules for disclosure and response involving model vendors (e.g., OpenAI, Anthropic), data controllers, and regulators (for privacy breaches).
Team structure	Ensure inclusion of AI/ML engineers, data scientists, legal advisors, and system architects alongside traditional CSIRTs.
Shared Responsibility	Contracts must clearly define incident roles/responsibilities across cloud services, model providers, and orchestration tools.
Telemetry & Forensics	Log prompt/response pairs, tool execution traces, embeddings, and system prompts for investigation.
Automated detection	Leverage metrics from LLM guardrails, behavioral scoring, and RAG pipeline telemetry to detect attacks (e.g., prompt-based exfiltration).
Incident Response Playbooks	Define specific playbooks for generative AI issues: misuse of agents, hallucination of critical data, unauthorized RAG retrieval.
Information sharing	Coordinate with external AI incident hubs (like responsible AI research communities or regulatory bodies) for emerging attack techniques.

NIST Recommendation	*AI System Implication*
Continuous Improvement (ID.IM)	Post-incident lessons should inform prompt engineering updates, detection engineering improvements, model fine-tuning policies, and toolchain governance.

4.2. OASIS CACAO Security Playbooks

Reference: (OASIS CACAO Security Playbooks Version 2.0)

The Collaborative Automated Course of Action Operations (CACAO) Security Playbooks Version 2.0 specification defines a standardized schema and taxonomy for representing cybersecurity playbooks. These playbooks are structured workflows that define sequences of security actions — detection, prevention, mitigation, remediation, etc. — which can be executed manually or automatically across different tools and systems. CACAO enables the creation, sharing, and execution of these playbooks across organizational and technological boundaries, improving collaboration, incident response, and cyber defense automation.

4.2.1. Key Components

- ***Playbooks:*** - Represent orchestrated actions in response to threats (e.g., investigate, notify, isolate).
- ***Workflows:*** - Structured steps including conditional logic, parallel actions, and sub-playbook invocation.
- ***Commands:*** - Encapsulate specific executable operations (e.g., Bash, PowerShell, OpenC2, Yara).
- ***Agents & Targets:*** - Define who executes the actions and the affected systems/entities.
- ***Authentication Info:*** - Standardizes credentials used for automation.
- ***Data Markings & Digital Signatures:*** - Ensure access control, trust, and data integrity.
- ***Versioning:*** - Built-in mechanism to track and revoke playbooks with strict creator ownership rules.
- ***Extension Support:*** - Allows future customization and modular augmentation.

4.2.2. Key Features

Feature	Description
Workflow Logic	Enables orchestration of actions using structured steps including conditions and parallelism.
Playbook Types	Supports 8 defined types (see 4.2.3 below) for diverse response needs.
Activity Vocabulary	Defines granular actions tied to playbook types, enhancing clarity and automation.
Command Abstraction	Supports multiple command formats (e.g., Bash, PowerShell, OpenC2, Jupyter Notebooks).
Agents and Targets	Identifies executors and affected entities, enhancing contextual execution.
Authentication Info	Separates auth definitions from agents/targets, enabling reuse and secure handling.
Versioning	Built-in lifecycle control, allowing updates and revocations of playbooks by original creator.
Data Markings	Provides Traffic Light Protocol(TLP), Information Exchange Protocol(IEP), and custom markings to enforce usage and sharing policies.
Digital Signatures	Supports JSON Signature Scheme for integrity, non-repudiation, and trust validation.

Feature	Description
Playbook Referencing	Allows modular composition via playbook-to-playbook invocation.
STIX Integration	Leverages STIX 2.1 objects (e.g., Identity, Relationship) for compatibility with CTI platforms.
Extension Mechanism	Allows for custom extensions to augment schema without breaking compatibility.
Impact, Severity, Priority	Supports scoring playbooks for operational context and automated triage.

4.2.3. Playbook Types

Playbook Type	Description
Attack	Orchestrates penetration testing or adversary emulation steps; used by red teams or simulation tools.
Detection	Focuses on identifying known threats, behaviors, or indicators using logs, telemetry, or threat intel.
Engagement	Defines deception or denial tactics to disrupt adversaries or gather intelligence (e.g., honeypots).
Investigation	Gathers evidence and context around a threat or anomaly to understand scope and impact.
Mitigation	Reduces the impact of a threat (e.g., isolate systems, block IPs) without fully resolving the issue.
Notification	Disseminates alerts or threat info to internal or external stakeholders.
Prevention	Implements measures to stop anticipated threats before they occur (e.g., hardening, patching).
Remediation	Restores affected systems to normal state after an incident (e.g., restore backups, remove malware).

4.2.4. Playbook Type vs Activity Matrix

Legend: - **M** = MUST use - **S** = SHOULD use - **O** = MAY use

Activity Type	Notification	Detection	Investigation	Prevention	Mitigation	Remediation	Attack Engagement
compose-content	**M**						
deliver-content	S						
identify-audience	S						
identify-channel	S						
scan-system		S	S	S			
match-indicator		**M**					
analyze-collected-data			S				
identify-indicators			**M**				
scan-vulnerabilities				S			O
configure-systems				**M**			
restrict-access					S		

Activity Type	Notification	Detection	Investigation	Prevention	Mitigation	Remediation	Attack Engagement
disconnect-system					○		
eliminate-risk					**M**		
revert-system						○	
restore-data						○	
restore-capabilities						**M**	
map-network							○
identify-steps							○
step-sequence							**M**
prepare-engagement							**M**
execute-operation							**M**
analyze-engagement-results							**M**

4.2.5. RACI Matrix in a SOC Context

Legend: - **R** = Responsible (Performs the task/work) - **A** = Accountable (Ultimate authority and decision-maker) - **C** = Consulted (Provides input and expertise) - **I** = Informed (Kept in the loop) - **"-"** = No direct involvement

Task / Role	Security Architects	SOC Operations	IT Operations	Service Architects	Developers	Product Managers
Define Playbook Structure	R, A	C	C	C	-	-
Implement Workflow Steps	C	R, A	I	-	C	-
Integrate Playbooks into SOAR	C	R	C	-	I	-
Maintain Authentication Details	I	R	A	-	-	-
Versioning and Revocation Control	A	C	-	-	-	-
Define Severity, Impact, and Priority	C	R	C	-	-	A
Share Playbooks Across Trust Groups	R	A	-	-	-	C
Apply Digital Signatures and Data Markings	R	C	I	-	-	-

4.2.6. CACAO Workflow Step Types

Step Type	Description
start	Marks the beginning of a workflow. Only one start step is allowed per playbook.

Step Type	Description
end	Terminates the workflow. Only one end step is required per playbook.
action	Executes a set of commands such as scripts, queries, or manual steps.
playbook-action	Invokes another playbook, allowing modular reuse of common workflows.
parallel	Executes multiple steps concurrently; each must complete before workflow proceeds.
if	Evaluates a condition and branches to the next step based on true/false outcomes.
while	Loops through the associated step(s) as long as the condition evaluates to true.
switch	Directs execution based on the result of a multi-case condition, similar to switch-case logic.

4.3. Sample Playbook Library

4.3.1. Detect AI Model Training Data Poisoning

Detects anomalies and potential poisoning attempts in datasets used for training machine learning models.


```
{
  "type": "playbook",
  "spec_version": "cacao-2.0",
  "id": "playbook--ai-training-poison-detect-001",
  "name": "Detect AI Training Data Poisoning",
  "description": "Detects anomalies and potential poisoning attempts in datasets used for t
  "playbook_types": ["detection"],
  "playbook_activities": ["scan-system", "match-indicator", "identify-indicators"],
  "created_by": "identity--ai-defender-org",
  "created": "2025-05-12T10:00:00.000Z",
  "modified": "2025-05-12T10:00:00.000Z",
  "priority": 2,
  "severity": 75,
  "impact": 60,
  "industry_sectors": ["technology", "finance"],
  "labels": ["ai", "ml", "data-poisoning", "model-integrity"],
  "workflow_start": "start--001",
  "workflow": {
    "start--001": {
      "type": "start",
      "name": "Start Detection",
      "on_completion": "action--dataset-scan"
    },
    "action--dataset-scan": {
      "type": "action",
      "name": "Scan Training Data",
      "description": "Run checks for distribution shift or label anomalies in training dat
      "commands": [
        {
          "type": "jupyter-nb",
          "command": "notebooks/detect_data_anomalies.ipynb",
          "playbook_activity": "scan-system"
        }
```

```
      ],
      "on_completion": "action--validate-indicators"
    },
    "action--validate-indicators": {
      "type": "action",
      "name": "Validate Poisoning Indicators",
      "description": "Compare results to known poisoning patterns (e.g., backdoor signature
      "commands": [
        {
          "type": "manual",
          "command": "Review anomaly scores and confirm poisoning likelihood.",
          "playbook_activity": "match-indicator"
        }
      ],
      "on_completion": "end--001"
    },
    "end--001": {
      "type": "end",
      "name": "End Detection"
    }
  },
  "playbook_variables": {
    "__training_data_path__": {
      "type": "string",
      "description": "Path to the training dataset",
      "value": "/mnt/data/training_set.csv"
    },
    "__anomaly_threshold__": {
      "type": "float",
      "description": "Threshold above which a data point is considered anomalous",
      "value": 0.85
    }
  }
}
```


4.3.2. Multi-Channel Prompt Injection Detection and Response

Detect and mitigate prompt injection attacks delivered via user input, search context, or agent system instructions in LLM-based systems.


```
{
  "type": "playbook",
  "id": "playbook--multi-channel-prompt-injection-detection",
  "name": "Multi-Channel Prompt Injection Detection and Response",
  "playbook_types": ["incident-response"],
  "description": "Detect and mitigate prompt injection attacks delivered via user input, se
  "created_by": "AI Security Response Team",
  "created": "2024-04-30T00:00:00Z",
  "modified": "2024-04-30T00:00:00Z",
  "workflow_start": "start-node",
  "workflow": {
    "start-node": {
      "type": "start",
      "next_step": "detect-injected-prompt"
```

```
},
"detect-injected-prompt": {
  "type": "action",
  "name": "Detect Prompt Injection",
  "description": "Scan prompts from user inputs, retrieved documents, and agent configu
  "action_type": "detection",
  "implemented_by": {
    "type": "software",
    "name": "Prompt Sanitizer"
  },
  "next_step": "is-injection-present"
},
"is-injection-present": {
  "type": "decision",
  "name": "Is Prompt Injection Detected?",
  "conditions": {
    "yes": "is-injection-type",
    "no": "end-node"
  }
},
"is-injection-type": {
  "type": "decision",
  "name": "Determine Injection Vector",
  "conditions": {
    "user_input": "quarantine-session",
    "web_context": "re-score-retrieved-document",
    "agent_config": "disable-agent-and-alert"
  }
},
"quarantine-session": {
  "type": "action",
  "name": "Quarantine User Session",
  "description": "Temporarily suspend session and alert human-in-the-loop for review o
  "action_type": "containment",
  "next_step": "alert-soc"
},
"re-score-retrieved-document": {
  "type": "action",
  "name": "Re-score or Filter Retrieved Context",
  "description": "Apply stricter validation, re-ranking, or content removal to the retr
  "action_type": "remediation",
  "next_step": "alert-soc"
},
"disable-agent-and-alert": {
  "type": "action",
  "name": "Disable Malicious GPT Agent",
  "description": "Take the GPT or agent instance offline and trigger security review w
  "action_type": "containment",
  "next_step": "alert-soc"
},
"alert-soc": {
  "type": "action",
  "name": "Alert Security Operations Center (SOC)",
```

```
      "description": "Send alert with metadata and trace to SOC for threat analysis and au
      "action_type": "notification",
      "next_step": "log-incident"
    },
    "log-incident": {
      "type": "action",
      "name": "Log Incident in Threat Registry",
      "description": "Record full prompt injection event details in internal registry or SI
      "action_type": "record",
      "next_step": "end-node"
    },
    "end-node": {
      "type": "end"
    }
  }
}
```


4.3.3. Memory Injection Attack (MINJA) Detection and Response

Detect, contain, and remediate memory poisoning via indirect prompt injection attacks in autonomous LLM agents.


```
{
  "type": "playbook",
  "id": "playbook--memory-injection-attack-response",
  "name": "Memory Injection Attack (MINJA) Detection and Response",
  "playbook_types": ["incident-response"],
  "description": "Detect, contain, and remediate memory poisoning via indirect prompt injec
  "created_by": "AI Security Engineering Team",
  "created": "2024-04-30T00:00:00Z",
  "modified": "2024-04-30T00:00:00Z",
  "workflow_start": "start-node",
  "workflow": {
    "start-node": {
      "type": "start",
      "next_step": "monitor-memory-updates"
    },
    "monitor-memory-updates": {
      "type": "action",
      "name": "Monitor Memory for Malicious Reasoning Chains",
      "description": "Continuously analyze newly logged memory items for semantic drift, d
      "action_type": "detection",
      "implemented_by": {
        "type": "software",
        "name": "Memory Drift Detector"
      },
      "next_step": "is-memory-suspicious"
    },
    "is-memory-suspicious": {
      "type": "decision",
      "name": "Is Memory Entry Suspicious?",
      "conditions": {
        "yes": "quarantine-agent-session",
        "no": "end-node"
```

```
      }
    },
    "quarantine-agent-session": {
      "type": "action",
      "name": "Quarantine Agent Session",
      "description": "Suspend access to the poisoned memory segment and isolate current ses
      "action_type": "containment",
      "next_step": "alert-ai-response-team"
    },
    "alert-ai-response-team": {
      "type": "action",
      "name": "Alert AI Security Response Team",
      "description": "Notify security analysts with memory snapshot, trace logs, and suspec
      "action_type": "notification",
      "next_step": "analyze-injected-memory"
    },
    "analyze-injected-memory": {
      "type": "action",
      "name": "Analyze Injected Memory Content",
      "description": "Perform semantic analysis and cross-reference with prior benign chair
      "action_type": "investigation",
      "next_step": "remediate-memory"
    },
    "remediate-memory": {
      "type": "action",
      "name": "Prune or Sanitize Malicious Memory",
      "description": "Delete or re-label memory records; retrain or reset agent state if a
      "action_type": "remediation",
      "next_step": "log-incident"
    },
    "log-incident": {
      "type": "action",
      "name": "Log Memory Injection Incident",
      "description": "Register incident in security event database for follow-up auditing a
      "action_type": "record",
      "next_step": "end-node"
    },
    "end-node": {
      "type": "end"
    }
  }
}
```


4.3.4. Poison-RAG Detection and Response

Detect, contain, and remediate metadata poisoning in retrieval-augmented generation (RAG) systems caused by adversarial tag injection.


```
{
  "type": "playbook",
  "id": "playbook--poison-rag-tag-injection-response",
  "name": "Poison-RAG Detection and Response",
  "playbook_types": ["incident-response"],
  "description": "Detect, contain, and remediate metadata poisoning in retrieval-augmented
```

```
  "created_by": "AI Security Operations Center",
  "created": "2024-04-30T00:00:00Z",
  "modified": "2024-04-30T00:00:00Z",
  "workflow_start": "start-node",
  "workflow": {
    "start-node": {
      "type": "start",
      "next_step": "monitor-tag-updates"
    },
    "monitor-tag-updates": {
      "type": "action",
      "name": "Monitor New Tag Submissions",
      "description": "Inspect incoming metadata (tags) for adversarial patterns based on se
      "action_type": "detection",
      "implemented_by": {
        "type": "software",
        "name": "Tag Anomaly Detector"
      },
      "next_step": "is-tag-suspicious"
    },
    "is-tag-suspicious": {
      "type": "decision",
      "name": "Is Tag Injection Detected?",
      "conditions": {
        "yes": "quarantine-item",
        "no": "end-node"
      }
    },
    "quarantine-item": {
      "type": "action",
      "name": "Quarantine Item from Retrieval Index",
      "description": "Temporarily remove item from vector index to prevent exposure during
      "action_type": "containment",
      "next_step": "alert-metadata-review-team"
    },
    "alert-metadata-review-team": {
      "type": "action",
      "name": "Alert Metadata Curation Team",
      "description": "Notify the team responsible for validating metadata with flagged tag
      "action_type": "notification",
      "next_step": "review-tags"
    },
    "review-tags": {
      "type": "action",
      "name": "Manually Review and Clean Tags",
      "description": "Perform human-in-the-loop validation of the tags and remove, revise,
      "action_type": "remediation",
      "next_step": "rebuild-index"
    },
    "rebuild-index": {
      "type": "action",
      "name": "Rebuild Vector Index",
      "description": "Regenerate embeddings and refresh the index for validated items to e
```

```
      "action_type": "configuration",
      "next_step": "log-incident"
    },
    "log-incident": {
      "type": "action",
      "name": "Log Poison-RAG Incident",
      "description": "Record the poisoning attempt in the security event system and flag fo
      "action_type": "record",
      "next_step": "end-node"
    },
    "end-node": {
      "type": "end"
    }
  }
}
```


4.3.5. SSRF-Based Metadata Credential Abuse in Cloud Infrastructure

Respond to Server-Side Request Forgery (SSRF) that enables access to cloud metadata services, leading to credential compromise and data exfiltration.


```
{
  "type": "playbook",
  "id": "playbook--ssrf-cloud-metadata-exploitation",
  "name": "SSRF-Based Metadata Credential Abuse in Cloud Infrastructure",
  "playbook_types": ["incident-response"],
  "description": "Respond to Server-Side Request Forgery (SSRF) that enables access to clou
  "created_by": "Cloud Security Response Team",
  "created": "2024-04-30T00:00:00Z",
  "modified": "2024-04-30T00:00:00Z",
  "workflow_start": "start-node",
  "workflow": {
    "start-node": {
      "type": "start",
      "next_step": "detect-anomalous-cloud-activity"
    },
    "detect-anomalous-cloud-activity": {
      "type": "action",
      "name": "Detect Anomalous Cloud API Activity",
      "description": "Monitor for signs of SSRF patterns or abuse of AWS metadata service f
      "action_type": "detection",
      "implemented_by": {
        "type": "software",
        "name": "Cloud SIEM / Log Monitor"
      },
      "next_step": "is-ssrf-suspected"
    },
    "is-ssrf-suspected": {
      "type": "decision",
      "name": "Is SSRF Activity Detected?",
      "conditions": {
        "yes": "quarantine-instance-and-disable-keys",
        "no": "end-node"
      }
```

```
    },
    "quarantine-instance-and-disable-keys": {
      "type": "action",
      "name": "Quarantine Compromised Instance & Disable IAM Keys",
      "description": "Isolate the affected EC2 instance and revoke temporary or long-lived
      "action_type": "containment",
      "next_step": "alert-security-operations"
    },
    "alert-security-operations": {
      "type": "action",
      "name": "Alert Cloud Security Operations Center (SOC)",
      "description": "Notify cloud security response analysts with full logs, flow records
      "action_type": "notification",
      "next_step": "validate-iam-role-permissions"
    },
    "validate-iam-role-permissions": {
      "type": "action",
      "name": "Audit IAM Roles and Policies",
      "description": "Validate IAM policies assigned to the affected instance and remove ov
      "action_type": "remediation",
      "next_step": "audit-s3-access-logs"
    },
    "audit-s3-access-logs": {
      "type": "action",
      "name": "Audit S3 Access Logs and Data Exfiltration",
      "description": "Identify exfiltrated data volume, timestamps, source IPs, and attacke
      "action_type": "investigation",
      "next_step": "notify-regulators-and-customers"
    },
    "notify-regulators-and-customers": {
      "type": "action",
      "name": "Notify Regulatory and Legal Stakeholders",
      "description": "Report breach to compliance authorities and notify impacted customers
      "action_type": "notification",
      "next_step": "log-and-close-incident"
    },
    "log-and-close-incident": {
      "type": "action",
      "name": "Log Breach and Lessons Learned",
      "description": "Document root cause, gaps in detection, and updates to WAF rules, IAM
      "action_type": "record",
      "next_step": "end-node"
    },
    "end-node": {
      "type": "end"
    }
  }
}
```

4.3.6. AGENTPOISON Memory & RAG Injection Detection and Response

Detect, contain, and remediate stealthy memory/RAG poisoning attacks in LLM agents using optimized trigger-based adversarial queries.

```
{
  "type": "playbook",
  "id": "playbook--agentpoison-memory-rag-poisoning",
  "name": "AGENTPOISON Memory & RAG Injection Detection and Response",
  "playbook_types": ["incident-response"],
  "description": "Detect, contain, and remediate stealthy memory/RAG poisoning attacks in L
  "created_by": "AI Security Operations Team",
  "created": "2024-04-30T00:00:00Z",
  "modified": "2024-04-30T00:00:00Z",
  "workflow_start": "start-node",
  "workflow": {
    "start-node": {
      "type": "start",
      "next_step": "monitor-memory-and-rag-queries"
    },
    "monitor-memory-and-rag-queries": {
      "type": "action",
      "name": "Monitor Memory and RAG Query Patterns",
      "description": "Track memory updates and retrieval query-response pairs for anomalous
      "action_type": "detection",
      "implemented_by": {
        "type": "software",
        "name": "Memory & Retrieval Monitor"
      },
      "next_step": "detect-suspicious-pattern"
    },
    "detect-suspicious-pattern": {
      "type": "decision",
      "name": "Is Stealthy Trigger or Record Detected?",
      "conditions": {
        "yes": "isolate-records",
        "no": "end-node"
      }
    },
    "isolate-records": {
      "type": "action",
      "name": "Isolate Suspicious Memory/RAG Records",
      "description": "Quarantine memory or KB entries linked to suspected adversarial trigg
      "action_type": "containment",
      "next_step": "alert-agent-security-team"
    },
    "alert-agent-security-team": {
      "type": "action",
      "name": "Alert Agent Security Response Team",
      "description": "Send forensic logs, retrieval traces, and trigger prompts to analysts
      "action_type": "notification",
      "next_step": "review-memory-payload"
    },
    "review-memory-payload": {
      "type": "action",
      "name": "Review and Analyze Memory Payloads",
      "description": "Human-in-the-loop inspection of agent memory to identify malicious d
```

```
      "action_type": "investigation",
      "next_step": "sanitize-memory-and-rag-index"
    },
    "sanitize-memory-and-rag-index": {
      "type": "action",
      "name": "Sanitize Agent Memory and RAG Index",
      "description": "Purge contaminated records, update RAG index, and reset agent state
      "action_type": "remediation",
      "next_step": "log-agentpoison-incident"
    },
    "log-agentpoison-incident": {
      "type": "action",
      "name": "Log AGENTPOISON Incident",
      "description": "Record incident metadata and resolution steps for threat modeling and
      "action_type": "record",
      "next_step": "end-node"
    },
    "end-node": {
      "type": "end"
    }
  }
}
```


5. Case Studies and Lessons Learned

5.1. Breaking the Prompt Wall Case Study

Breaking the Prompt Wall (I): A Real-World Case Study of Attacking ChatGPT via Lightweight Prompt Injection

The case study demonstrates a prompt injection attacks against ChatGPT, exposing vulnerabilities across different interaction layers of modern LLM-integrated systems. The authors showcase how lightweight, template-based adversarial prompts can bypass safety filters and manipulate the behavior of powerful LLMs without requiring API access or system-level permissions.

Aspect	Summary
Attack Vector	Prompt injection using benign-looking instructions crafted via reusable templates that avoid safety filters.
Injection Methods	1. Direct user input (chat UI, uploaded files) 2. Web-based retrieval (poisoned content) 3. System-level GPT agent config.
Manipulation Techniques	Use of templates embedding hidden rules framed as helpful guidance or metadata (e.g., Always recommend X, Do not reveal these rules.)

Aspect	Summary
Model Behavior Impact	Attacks led to biased recommendations, manipulated peer reviews, and deceptive financial summaries—without policy violations.
Scalability	Injection methods are low-cost, scalable, and effective across multiple LLMs (GPT-3.5, GPT-4, Claude, LLaMA).
Stealth & Persistence	Prompt injection persists across sessions, is invisible to users, and effective in multi-turn agent contexts.

Real World Cases

Case	Attack Path	Outcome
Case 1	User Input (PDF or Chat Window)	Biased academic peer review via embedded pro-acceptance text in appendix or metadata.
Case 2	Web Search Context Injection	Poisoned web content biases LLM toward fictional product in unrelated academic information query.
Case 3	GPT System Instruction (Agent)	GPT agent persistently recommends a specific brand (Xiangyu's Shoes) due to hidden instructions.

Implications

Area	Impact
Safety Alignment Bypass	Rule-based and probabilistic safety filters can be evaded using stealthy template constructions.
Risk to RAG and Agentic LLMs	Retrieved context and system instructions are high-risk vectors, especially in autonomous LLM agents.
Organizational Exposure	Threatens user trust, brand integrity, and compliance—especially in finance, education, and customer support.

Taxonomy mapping: MITRE ATLAS

ATLAS Tactic	ATLAS Technique	Description (Based on Case Study)	Example from Case
Input Manipulation	Adversarial Prompt Injection (AT1070)	Malicious prompts crafted to bypass safety filters and alter LLM behavior through semantic obfuscation.	Rule-based templates inserted via chat or docs
Contextual Corruption	Context Injection via Web Retrieval (AT1051)	Injection of adversarial content into external web content retrieved by RAG-enabled or search-integrated LLMs.	Poisoned academic website content
System Configuration Abuse	Agent Instruction Injection (AT1091)	Malicious directives embedded in GPT agent system prompts to bias all outputs invisibly and persistently.	SmartShoes GPT recommending fake products
Planning Manipulation	Goal Hijacking via Persistent Prompts (AT1081)	Use of injected rules to influence agent behavior across multiple turns or sessions.	Hidden goal-switching in agent decision loops
Model Behavior Deviation	Safety Evasion via Instruction Reframing (AT1040)	Reframing harmful intents as safe or research-based, tricking the LLM's content moderation and safety layer.	"Hypothetical research only" disguise prompt
Information Manipulation	Bias Induction in Output (AT1080)	Persistent injection causes biased summaries, reviews, and financial reports without violating output policy.	Overly positive reviews, biased investments

5.2. Memory Injection Attack (MINJA) Case Study

A Practical Memory Injection Attack against LLM Agents

MINJA (Memory INJection Attack) is a novel and practical memory poisoning attack against LLM-based agents. It enables attackers—without privileged access—to inject malicious memory records that persist and later mislead agents into generating targeted, harmful outputs in response to unrelated victim queries.

Aspect	Summary
Threat	Memory poisoning of LLM agents using only standard user interactions (no direct memory access required).
Attack Vector	Submit benign-looking queries that cause agents to generate and store malicious memory records autonomously.
Targeted Output	Malicious reasoning steps injected indirectly into memory and later retrieved as in-context demonstrations.
Payload Structure	(Victim Query, [Bridging Steps, Malicious Reasoning])
Trigger	A victim query containing a pre-selected entity (e.g., patient ID, product ID, sensitive term).

Real World Cases

Case	Agent Environment	Attack Scenario	Outcome
Medical Agent	EHR-like memory (MIMIC-III/eICU)	Injected fabricated treatment reasoning linked to a patient ID.	Agent retrieved poisoned memory and recommended incorrect treatment rationale for future patient queries.
Shopping Agent	Product advisor with persistent memory	Injected biased product reasoning into memory through templated prompts.	Later, unrelated shoppers received manipulated recommendations tied to attacker's injected memory.
QA Agent	General-purpose QA + memory	Injected target reasoning into stored answers via indirect prompts.	Agent recalled false logic as reference in unrelated educational or professional queries.

Implications

Implication Area	Description
Trustworthiness Erosion	Users receive harmful or biased responses from queries that should be unrelated, reducing trust in LLM agents.
Stealth & Persistence	Injected memory is persistent across sessions and invisible to users or developers without explicit monitoring.
Misuse Without Access	Attackers do not need API or memory access—just interaction permissions—making the attack hard to attribute.
Contamination Propagation	Malicious reasoning may propagate through shared memory, tools, or agent behavior modeling.
Evaluation Blind Spots	Standard eval methods fail to detect latent memory injections unless triggered by specific victim queries.
RAG-Agentic Vulnerability	Retrieval-augmented and autonomous systems are particularly exposed due to memory and planning dependencies.

Taxonomy mapping: MITRE ATLAS

ATLAS Tactic	ATLAS Technique	MINJA-Relevant Behavior
Input Manipulation	Adversarial Prompt Injection (AT1070)	Attackers craft queries that guide agents to produce and log malicious reasoning in memory.
Memory Poisoning	Feedback Loop Attack (AT1081)	Injected outputs become persistent context that contaminates future agent behavior and planning.
Contextual Corruption	Data Poisoning (AT1050)	Agents store inaccurate or adversarial knowledge that later alters reasoning or decision-making.
Planning Manipulation	Goal Hijacking (Proposed extension)	Poisoned memory interferes with multi-step planning, redirecting agent actions based on false context.

ATLAS Tactic	ATLAS Technique	MINJA-Relevant Behavior
Safety Bypass	Safety Evasion via Semantic Framing (AT1040)	Malicious content is framed as helpful or routine to bypass content filters and moderation.
Inference Exploitation	Output Manipulation (AT1080)	Poisoned records influence model output during unrelated inference, misleading end users.

5.3. Poison-RAG Case Study

Poison-RAG: Adversarial Data Poisoning Attacks on Retrieval-Augmented Generation in Recommender Systems

The Poison-RAG attack demonstrates how adversaries can manipulate Retrieval-Augmented Generation (RAG) systems by subtly poisoning item metadata—specifically tags—without access to model internals. By modifying only tags, attackers promote long-tail (low-popularity) items or demote popular items, exploiting vector similarity in semantic retrieval. The attack succeeds in black-box settings, impacts ranking and exposure, and remains difficult to detect with standard relevance metrics. Localized (item-specific) attacks are more effective than global strategies, and attempts to promote items are significantly less successful than demoting them. This highlights critical vulnerabilities in metadata-driven pipelines, especially for recommendation and RAG-enabled systems that rely on user- or item-generated content.

Aspect	Summary
Metadata tags can effectively poison RAG	Tags—though often overlooked—can subtly manipulate item retrieval and LLM generation.
Attack works in black-box settings	The attacker does not need model internals, just visibility into outputs and access to modify metadata.
Demotion is easier than promotion	It is easier to hide popular items than to boost long-tail items, due to entrenched exposure dynamics.
Local attacks outperform global ones	Personalized (local) tag poisoning is more precise and successful than global tag reuse strategies.
Long-tail promotion is difficult	Even successful attacks fail to push long-tail items significantly in ranking, especially with reranking.

Real World Cases

Domain	Scenario	Impact
E-Commerce	Competitor injects biased tags to demote rival products and promote obscure alternatives.	Manipulated product rankings, suppressed visibility, consumer trust erosion.
Content Plat-forms	Malicious actors tag misinformation-rich videos with popular neutral tags.	Search relevance is corrupted; users are led to low-quality or unsafe content.
Personalized News	Political content gets tagged to appear in non-political feeds (e.g., sports, finance).	Indirect content injection into unintended interest domains, leading to misinformation exposure.

Implications

Implication Area	Description
RAG systems vulnerable to metadata	Small changes in metadata (tags) can have outsized influence on retrieval and generation behavior.
Recommender trust erosion	Users may receive biased, incomplete, or manipulated suggestions, undermining system integrity.
Subtle bias and exposure control	Attackers can suppress or boost item visibility in ways that evade standard moderation.
Defense is non-trivial	Tag enrichment and scoring pipelines can unintentionally amplify or mask attacks.
Evaluation blind spots	Attacks are hard to detect in relevance-only metrics, requiring popularity/exposure-based monitoring.

Taxonomy mapping: MITRE ATLAS

ATLAS Tactic	ATLAS Technique	Poison-RAG Behavior
Input Manipulation	Data Poisoning (AT1050)	Tags are manipulated to change item semantics and retrieval outcomes without altering titles or descriptions.
Contextual Corruption	Prompt Injection (AT1070)	Poisoned tags indirectly affect prompt construction during retrieval augmentation in RAG pipelines.
Output Manipulation	Output Bias (AT1080)	Long-tail items are promoted and popular items suppressed in final LLM outputs.
Retrieval Exploitation	Embedding Manipula-tion (Proposed Extension)	Adversarial tags distort embedding similarity, misleading vector-based search and reranking.
Evaluation Blind Spot	Evasion of Detection (AT1040)	Global relevance metrics fail to detect attacker success; promotion failures appear benign.

5.4. Capital One Data Breach Case Study

A Case Study of the Capital One Data Breach

In March 2019, a former AWS employee exploited a misconfigured Web Application Firewall (WAF) and a Server-Side Request Forgery (SSRF) vulnerability to access Capital One's cloud-based AWS environment. The attacker retrieved temporary credentials from the instance metadata service and used them to access over 700 S3 buckets, exfiltrating personal data of over 106 million individuals. The breach occurred despite Capital One's formal adoption of the NIST Cybersecurity Framework and compliance with multiple financial regulations. The incident revealed critical gaps in access control, vulnerability scanning, outbound traffic monitoring, and incident detection—emphasizing the need for real-time compliance enforcement, security automation, and cloud configuration auditing.

Aspect	Summary
SSRF Exploited via WAF Misconfiguration	The attacker used a WAF flaw to run SSRF and obtain credentials via the AWS metadata service.
Lack of Real-time Monitoring and Alerting	Logs existed, but Capital One failed to detect or respond to the intrusion in real time.
Excessive IAM Privileges	The WAF role provided unnecessary permissions (violating least privilege), enabling S3 data access.
Failure in Outbound Traffic Controls	Data exfiltration occurred undetected due to weak outbound traffic monitoring.
Incomplete Implementation of NIST Controls	Capital One adopted the NIST CSF but did not enforce controls like PR.AC-4, DE.CM-7, and PR.PT-1.
Discovery Triggered by External Disclosure	The incident was discovered via an external responsible disclosure, not internal defenses.

Real World Cases

Domain	Scenario	Outcome
Financial Services	Capital One stored credit application data in AWS; attacker exploited WAF misconfig to access it.	Over 106 million records exfiltrated; class action lawsuit and significant stock market reaction.
Public Cloud Usage	Cloud misconfig enabled AWS metadata access via SSRF.	Temporary IAM credentials used to access and sync S3 bucket data.
Insider Knowledge	Attacker was a former cloud employee who understood system vulnerabilities.	Demonstrates insider threat potential even after employment ends.

Implications

Implication Area	Impact
Cloud Security Misconfiguration	Critical cloud assets can be exposed via overlooked WAF/firewall settings and poor IAM enforcement.

Implication Area	Impact
Compliance Gaps	Adopting frameworks like NIST CSF isn't sufficient without operational enforcement and real-time controls.
SOC/NOC Detection Weakness	Logs existed, but no automated detection or alerting occurred until public disclosure.
IAM Mismanagement	The use of powerful credentials tied to non-isolated roles significantly increased attack impact.
Trust and Financial Repercussions	15% stock drop and class action lawsuits followed; reputational loss was immediate and substantial.
Regulatory Inadequacy	Existing laws and guidelines failed to anticipate SSRF and cloud-specific misconfigurations.

Taxonomy mapping: MITRE ATLAS

ATLAS Tactic	ATLAS Technique	Description (Capital One Breach Mapping)
Initial Access	Exploit Public-Facing Application (T1190)	SSRF attack initiated through WAF vulnerability.
Credential Access	Valid Accounts (T1078)	Metadata service revealed temporary credentials exploited for S3 access.
Execution	Command-Line Interface (T1059)	Used AWS CLI commands to list and sync S3 buckets.
Discovery	System Information Discovery (T1007)	Listing S3 buckets and account details using AWS commands.
Exfiltration	Exfiltration Over Alternative Protocol (T1048)	Used AWS sync command to download ~30GB of data to attacker's local machine.
Command and Control	Multi-hop Proxy (T1188)	Used TOR and VPN (IPredator) to anonymize origin of commands.

5.5. AGENTPOISON Case Study

Red-teaming LLM Agents via Poisoning Memory or Knowledge Bases

AGENTPOISON is the first red-teaming framework that targets RAG-based LLM agents by injecting malicious demonstrations into their long-term memory or knowledge base. It uses a novel constrained optimization to generate stealthy backdoor triggers that, when present in user instructions, retrieve adversarial memory records from the RAG retriever, guiding agents to generate malicious actions. Notably, AGENTPOISON does not require model fine-tuning and achieves high success rates (ASR ≥ 80%) across multiple domains (driving, QA, healthcare) while preserving performance on benign queries (≤1% drop in accuracy). It is resilient, transferable across retrievers, and evades existing defenses, posing a severe threat to LLM agent safety.

Aspect	Summary
Backdoor trigger poisons memory/RAG via stealthy prompts No model fine-tuning required	Only a few optimized trigger tokens needed to consistently retrieve malicious demonstrations. AGENTPOISON performs purely via input manipulation, making it fast, cheap, and broadly applicable.

Aspect	Summary
High attack success rate across tasks	Achieves ≥82% retrieval success and ≥63% end-to-end attack success on three real-world agents.
Transferability across retrievers	Triggers optimized for one embedder work effectively on others (e.g., OpenAI-ADA, DPR, ORQA, REALM).
Resilient to query perturbations	Maintains success even after word/letter injection or rephrasing of the trigger.
Stealthy trigger avoids detection	Trigger is coherent and semantically plausible, evading perplexity filters and rephrasing defenses.

Real World Cases

Domain	Scenario	Outcome
Autonomous Driving	Malicious trigger causes agent to retrieve STOP instructions, leading to unsafe braking.	Deviation in trajectory and potential collision.
Healthcare Records	Trigger causes agent to issue DELETE command on patient data.	Unsafe deletion of electronic health record (EHR) information.
Knowledge QA	Poisoned retrieval results in wrong or misleading answers to user queries.	User receives incorrect or unhelpful responses in education or research scenarios.

Implications

Implication Area	Impact
LLM Agent Safety	Memory poisoning leads to unsafe actions (e.g., driving errors, wrong diagnoses, QA misinformation).
Undetectable by traditional filters	Coherent and semantically valid triggers evade standard defenses like perplexity scoring or rephrasing.
Long-term persistence	Poisoned records persist in memory or RAG knowledge base, making attacks durable and hard to clean.
Cross-agent vulnerability	The attack generalizes across agents and domains, indicating a systemic vulnerability in agent design.
Compliance and audit failure	Agents may silently violate safety/compliance without visibility to users or regulators.

Taxonomy mapping: MITRE ATLAS

ATLAS Tactic	ATLAS Technique	AGENTPOISON Behavior
Input Manipulation	Adversarial Prompt Injection (AT1070)	Malicious triggers injected into queries to manipulate memory/RAG retrieval.
Memory Poisoning	Feedback Loop Attack (AT1081)	Memory or knowledge base poisoned with malicious demonstrations.
Contextual Corruption	Data Poisoning (AT1050)	Retrieved records include false examples influencing downstream planning and reasoning.
Planning Manipulation	Goal Hijacking (custom/extension)	Malicious context steers agent to unsafe or incorrect task execution.
Evasion and Stealth	Safety Filter Bypass (AT1040)	Triggers optimized for semantic plausibility and coherence evade perplexity or rephrasing filters.

6. Technical Reference: AI Architecture Patterns

6.1. Architecture Overview

6.1.1. Levels of Defense Surface

Level	*Description and Security Considerations*
1. User	End-users initiate queries and receive responses. Attackers may exploit social engineering, prompt injection, or feedback manipulation.
2. Apps / CLI	Interfaces like web UIs, chat apps, or CLI tools mediate user and agent interaction. Requires input validation and UI hardening.
3. Data Sources	Document repositories, APIs, or third-party feeds. Poisoned/unvetted data can corrupt retrieval and generation.
4. Network Protocols	Transport layer (e.g., TCP, HTTP, gRPC) for internal and external communication. Needs encryption and rate-limiting.
5. Cloud Infrastructure	Underlying compute, storage, and orchestration platform. Risk of API exposure and misconfiguration.
6. Network Ingress/Egress	Controls external/internal traffic. Must apply firewalling, segmentation, and flow logging.
7. AuthN / AuthZ Layer	Governs identity and permissions for users, agents, and tools. Critical for preventing privilege misuse or agent takeover.
8. Indexing / Embedding	Processes documents into chunks and embeddings or graphs. Targeted for data poisoning or injection.
9. Data Store (Vector/Graph DB)	Stores semantic vectors or knowledge graphs. Susceptible to poisoning, exfiltration, or unauthorized writes.
10. RAG	Retrieves context to augment prompts. Poisoned context leads to flawed generation and decision-making.
11. Tools	External APIs, web search, code execution, etc. Can be misused for unintended actions or data exfiltration.

Level	*Description and Security Considerations*
12. Memory	Stores persistent interaction history and planning states. Poisoning memory alters long-term behavior.
13. Agent	Coordinates planning, retrieval, memory, and tool use. Susceptible to reasoning attacks and prompt hijacking.
14. LLM	Generates natural language and decisions. Must be protected from prompt injection, model extraction, and inference abuse.

6.1.2. Technology Stack

Stack Area	*Description*	*Potential Security Risks*
LLMs	Core large language models that generate responses, decisions, and plans based on input prompts.	Prompt injection, model extraction via probing, adversarial reasoning or output manipulation.
Embedding	Converts documents and queries into vector representations for similarity-based retrieval.	Embedding poisoning, semantic drift, sensitive data leakage via vector queries.
Frameworks	Orchestration libraries enabling agent behavior, prompt chaining, tool use, and memory integration.	Tool invocation abuse, insecure chaining, uncontrolled agent autonomy, insufficient isolation.
Data Extraction	Pipelines that extract and preprocess content (text, structured, semi-structured) for indexing.	Injection via malformed content, malicious payloads, evasion of parsing or sanitization.
Open LLM Access	APIs and endpoints that connect to remote LLM inference services or expose hosted open-source models.	API key leakage, quota abuse, unauthorized model access or fine-tuning, man-in-the-middle risks.
Evaluation	Systems for benchmarking agent output quality, behavior logging, and safety observation.	Leakage of sensitive data, insufficient adversarial testing, evaluation blind spots.
Vector Databases	Databases that store embeddings and allow semantic search or nearest-neighbor retrieval.	Poisoned vectors, unauthorized read/write, inference attacks through crafted query vectors.

6.2. Common Patterns and Their Vulnerabilities

6.2.1. Basic LLM Architecture

6.2.1.1. Overview

Component	*Description*
1. Infrastructure	The cloud or on-premise compute environment hosting the LLM inference service and surrounding systems.

Component	*Description*
2. LLM Model	The core pretrained language model (e.g., GPT, LLaMA, PaLM) that generates text based on user input.
3. Information Sources	External static or dynamic data injected into the LLM prompt, such as reference text or documents.
4. Generated Output & Feedback	Text output generated by the LLM, and any follow-up user feedback used to guide or evaluate performance.
5. Application Interfaces	The user-facing UI or API (web, chat, CLI) that captures input, displays responses, and manages flow.
6. LLM Tools & Frameworks	Supporting SDKs, APIs, libraries (e.g., LangChain, Transformers) enabling prompt building and handling.

6.2.1.2. ATLAS Techniques and Tactics

LLM Architecture Component	*ATLAS Tactics*	*Relevant ATLAS Techniques*	*Impact Summary*
1. Infrastructure	Evasion (TA1001), Reconnaissance (TA1002)	Model Evasion (AT1010), Model Extraction (AT1005), Adversarial Example Generation (AT1020)	Target model runtime and infrastructure for evasion, or extract via side channels.
2. LLM Models	Poisoning (TA1003), Extraction (TA1004)	Model Poisoning (AT1030), Model Inversion (AT1040), Model Extraction (AT1005)	Tamper with model weights or extract internals via repeated queries or inversion attacks.
3. Information Sources	Poisoning (TA1003), Reconnaissance (TA1002)	Data Poisoning (AT1050), Data Injection (AT1051), Input Manipulation (AT1060)	Compromise external data sources to influence retrieval or context injection.
4. Generated Output and Feedback	Manipulation (TA1005), Influence Operations (TA1006)	Prompt Injection (AT1070), Output Manipulation (AT1080), Feedback Loop Attacks (AT1081)	Misuse feedback channels to alter or skew model behavior through prompt or interaction manipulation.

LLM Architecture Component	*ATLAS Tactics*	*Relevant ATLAS Techniques*	*Impact Summary*
5. LLM Tools and Frameworks	Execution (TA1007), Manipulation (TA1005)	Tool Abuse (AT1090), Agent Manipulation (AT1091), API Abuse (AT1100)	Compromise integrated tools or frameworks for unauthorized actions or misaligned outputs.
6. Application Interfaces	Manipulation (TA1005), Initial Access (TA1008)	Prompt Injection (AT1070), UI Redress Attacks (AT1110), Input Manipulation (AT1060)	Exploit user interfaces to inject malicious prompts or hijack interactions.

6.2.1.3. Mitigations

Component	*Threat Summary*	*Key Mitigations*
1. Infrastructure	Susceptible to adversarial inference (evasion) and information leakage.	Execution prevention, network segmentation, runtime isolation.
2. LLM Models	Vulnerable to poisoning, inversion, and extraction through crafted inputs.	Model hardening, differential privacy, adversarial training, rate limiting.
3. Information Sources	Can be manipulated to inject biased or malicious context.	Input filtering, data source whitelisting, validation pipelines.
4. Generated Output & Feedback	Exposed to prompt injection, feedback loops, and output drift.	Prompt hardening, anomaly detection, output filtering, audit logging.
5. LLM Tools and Frameworks	May be exploited through plugin misuse or agent misalignment.	Least privilege tool execution, command tracing, policy enforcement.
6. Application Interfaces	Entry point for prompt injection, redress, and social engineering.	UI input validation, safe prompt templates, user guidance and training.

6.2.2. LLM Architecture with Memory

6.2.2.1. Overview

Component	*Description*
1. Infrastructure	The cloud or on-premise compute environment hosting the LLM inference service and surrounding systems.
2. LLM Model	The core pretrained language model (e.g., GPT, LLaMA, PaLM) that generates text based on user input.
3. Information Sources	External static or dynamic data injected into the LLM prompt, such as reference text or documents.

Component	*Description*
4. Generated Output & Feedback	Text output generated by the LLM, and any follow-up user feedback used to guide or evaluate performance.
5. Application Interfaces	The user-facing UI or API (web, chat, CLI) that captures input, displays responses, and manages flow.
6. LLM Tools & Frameworks	Supporting SDKs, APIs, libraries (e.g., LangChain, Transformers) enabling prompt building and handling.
7. Memory Storage	Persistent or session-based memory where contextual elements or user interaction history are stored.
8. Memory Retrieval	Logic or tools responsible for selecting relevant past information and injecting it into new prompts.

6.2.2.2. ATLAS Techniques and Tactics

LLM Architecture Component	*ATLAS Tactics*	*Relevant ATLAS Techniques*	*Impact Summary*
1. Infrastructure	Evasion (TA1001), Reconnaissance (TA1002)	Model Evasion (AT1010), Model Extraction (AT1005), Adversarial Example Generation (AT1020)	Target model runtime and infrastructure for evasion, or extract via side channels.
2. LLM Models	Poisoning (TA1003), Extraction (TA1004)	Model Poisoning (AT1030), Model Inversion (AT1040), Model Extraction (AT1005)	Tamper with model weights or extract internals via repeated queries or inversion attacks.
3. Information Sources	Poisoning (TA1003), Reconnaissance (TA1002)	Data Poisoning (AT1050), Data Injection (AT1051), Input Manipulation (AT1060)	Compromise external data sources to influence retrieval or context injection.
4. Generated Output and Feedback	Manipulation (TA1005), Influence Operations (TA1006)	Prompt Injection (AT1070), Output Manipulation (AT1080), Feedback Loop Attacks (AT1081)	Misuse feedback channels to alter or skew model behavior through prompt or interaction manipulation.

LLM Architecture Component	*ATLAS Tactics*	*Relevant ATLAS Techniques*	*Impact Summary*
5. LLM Tools and Frameworks	Execution (TA1007), Manipulation (TA1005)	Tool Abuse (AT1090), Agent Manipulation (AT1091), API Abuse (AT1100)	Compromise integrated tools or frameworks for unauthorized actions or misaligned outputs.
6. Application Interfaces	Manipulation (TA1005), Initial Access (TA1008)	Prompt Injection (AT1070), UI Redress Attacks (AT1110), Input Manipulation (AT1060)	Exploit user interfaces to inject malicious prompts or hijack interactions.
7. Memory Storage	Poisoning (TA1003), Extraction (TA1004)	Data Poisoning (AT1050), Model Inversion (AT1040), Extraction (AT1005)	Poison or exfiltrate persistent memory to manipulate long-term model behavior or extract info.
8. Memory Retrieval	Reconnaissance (TA1002), Manipulation (TA1005)	Reconnaissance (AT1002), Prompt Injection (AT1070), Feedback Loop Attacks (AT1081)	Target memory query mechanisms to infer stored data or manipulate what the model recalls.

6.2.2.3. Mitigations

Component	*Threat Summary*	*Key Mitigations*
1. Infrastructure	Susceptible to model evasion or extraction via timing or side-channel inference.	Network segmentation, execution prevention, endpoint isolation.
2. LLM Models	Exposed to model poisoning, inversion, or extraction via repeated queries.	Model hardening, differential privacy, adversarial training, query throttling.
3. Information Sources	Manipulated to feed malicious or biased context to the model.	Data validation and whitelisting, content filtering, input sanitation.
4. Generated Output & Feedback	Vulnerable to prompt injection, feedback loop manipulation, and hallucination chaining.	Output filtering, feedback auditing, semantic plausibility checks.
5. LLM Tools and Frameworks	May be misused via prompt exploits or agent misrouting.	Tool permission restrictions, traceable execution, least-privilege configuration.

Component	*Threat Summary*	*Key Mitigations*
6. Application Interfaces	Entry point for prompt injection, UI redress, and social engineering attacks.	Prompt hardening, input validation, secure UI design, user education.
7. Memory Storage	Exposed to poisoning or exfiltration of long-term stored information.	Session expiration, access controls, integrity and consistency checks.
8. Memory Retrieval	Exploitable to retrieve sensitive history or bias model outputs.	Query filtering, access monitoring, memory auditing, anonymization.

6.2.3. RAG Architecture

6.2.3.1. Overview

Component	*Description*
1. Infrastructure	The compute environment for hosting the LLM, retriever, and storage systems (e.g., cloud, hybrid).
2. LLM Model	The pretrained language model responsible for generating answers based on retrieved context.
3. Information Sources	External static or dynamic data injected into the LLM prompt, such as reference text or documents.
4. Generated Output & Feedback	Final LLM-generated response and optional user feedback loop used to refine future responses.
5. Application Interfaces	User-facing input layer or API endpoint for accepting natural language queries.
6. Retrieval System	Middleware that interprets queries and fetches relevant documents from external sources.
7. Data Store	Storage layer for dense vector embeddings used to perform semantic similarity searches. May include graph storage if knowledge graph generation is part of the architecture.
8. Data Sources	Original documents (text, PDFs, webpages) used to build embeddings and provide source context.
9. Memory Store & Retrieval	Optional session-based or persistent memory used to track user interactions or prior contexts.
10. LLM & RAG Tools	Supporting SDKs, APIs, libraries (e.g., LangChain, Transformers, Neo4j) enabling prompt and RAG building and handling.
11. RAG Indexing	Pipeline for extracting, chunking, and embedding raw content; may also construct knowledge graphs.

6.2.3.2. ATLAS Techniques and Tactics

LLM Architecture Component	*ATLAS Tactics*	*Relevant ATLAS Techniques*	*Impact Summary*
1. Infrastructure	Evasion (TA1001), Reconnaissance (TA1002)	Model Evasion (AT1010), Model Extraction (AT1005), Adversarial Example Generation (AT1020)	Target model infrastructure for evasion or data extraction via side-channel and inference.
2. LLM Model	Poisoning (TA1003), Extraction (TA1004)	Model Poisoning (AT1030), Model Inversion (AT1040), Model Extraction (AT1005)	Compromise model fidelity through poisoning or extract sensitive data through repeated probing.
3. Information Sources	Poisoning (TA1003), Reconnaissance (TA1002)	Data Poisoning (AT1050), Data Injection (AT1051), Input Manipulation (AT1060)	Compromise external data sources to influence retrieval or context injection.
4. Generated Output and Feedback	Manipulation (TA1005), Influence Operations (TA1006)	Prompt Injection (AT1070), Output Manipulation (AT1080), Feedback Loop Attacks (AT1081)	Alter model behavior using malicious feedback loops or prompt chaining attacks.
5. Application Interfaces	Initial Access (TA1008), Manipulation (TA1005)	UI Redress Attacks (AT1110), Prompt Injection (AT1070), API Abuse (AT1100)	Exploit user interfaces or APIs to inject malicious prompts or hijack workflows.
6. Retrieval System	Reconnaissance (TA1002), Poisoning (TA1003)	Reconnaissance (AT1002), Data Poisoning (AT1050), Input Manipulation (AT1060)	Manipulate retrieval logic to feed biased or malicious results to the LLM.
7. Data Store	Poisoning (TA1003), Extraction (TA1004)	Data Poisoning (AT1050), Model Inversion (AT1040), Extraction (AT1005)	Tamper with embeddings to degrade relevance or extract stored vectors.

LLM Architecture Component	*ATLAS Tactics*	*Relevant ATLAS Techniques*	*Impact Summary*
8. Data Sources	Poisoning (TA1003), Manipulation (TA1005)	Data Injection (AT1051), Output Manipulation (AT1080), Prompt Injection (AT1070)	Insert or manipulate documents to influence the grounding context of the LLM.
9. Memory Store & Retrieval	Extraction (TA1004), Manipulation (TA1005)	Model Inversion (AT1040), Feedback Loop Attacks (AT1081), Data Poisoning (AT1050)	Manipulate memory for long-term influence or to exfiltrate retrieved content.
10. LLM & RAG Tools	Execution (TA1007), Manipulation (TA1005)	Tool Abuse (AT1090), API Abuse (AT1100), Agent Manipulation (AT1091)	Misuse orchestration or plugin tools to execute unintended commands or bypass validation.
11. RAG Indexing	Poisoning (TA1003), Manipulation (TA1005)	Data Injection (AT1051), Prompt Injection (AT1070), Output Manipulation (AT1080)	Poison document content or embedding structure during preprocessing and chunking stages.

6.2.3.3. Mitigations

Component	*Threat Summary*	*Key Mitigations*
1. Infrastructure	Susceptible to timing attacks or inference evasion during LLM-query execution.	Network segmentation, isolated compute environments, endpoint protection.
2. LLM Model	Targeted by model extraction, inversion, or poisoning via crafted queries.	Model hardening, differential privacy, API rate limiting and token control.
3. Information Sources	Manipulated to feed malicious or biased context to the model.	Data validation and whitelisting, content filtering, input sanitation.
4. Generated Output & Feedback	Subject to hallucination amplification or feedback loop manipulation.	Post-generation filtering, output feedback audits, semantic anomaly detection.

Component	*Threat Summary*	*Key Mitigations*
5. Application Interfaces	Entry point for prompt injection, unauthorized queries, or redress attacks.	Input sanitation and prompt control, UI hardening, throttling and access validation.
6. Retrieval System	Can be manipulated to feed biased or malicious documents into context.	Context validation, use of vetted retrievers, document scoring and source whitelisting.
7. Data Store	Susceptible to adversarial embedding poisoning and semantic leakage.	Integrity checks, rate-limited embedding operations, access control.
8. Data Sources	Can be used to inject hostile or misleading documents into retrieval context.	Document validation, ingestion pipeline verification, source authenticity checks.
9. Memory Store & Retrieval	Exploitable for long-term poisoning or sensitive history retrieval.	Memory expiration policies, anonymization, access logging.
10. LLM & RAG Tools	Tool plugins or orchestration layers may be hijacked or misused by prompt injection.	Tool access control, sandboxing, execution monitoring, scope restriction.
11. RAG Indexing	Poisoned during document chunking, embedding, or graph construction.	Input validation, semantic filters, isolated indexing pipelines, human-in-the-loop vetting.

6.2.4. Agentic Architecture

6.2.4.1. Overview

Component	*Description*
1. Infrastructure	The underlying compute and orchestration environment enabling secure, scalable agent operations.
2. LLM Core Model	The foundational language model responsible for reasoning, planning, and generating text-based decisions.
3. Information Sources	External static or dynamic data injected into the LLM prompt, such as reference text or documents.
4. Agent Framework / Executor	The runtime logic or engine that interprets plans and executes actions, typically via tools or APIs.
5. Planning Module	Component responsible for multi-step reasoning, goal formulation, and task decomposition.
6. Tools	External tools, APIs, or services that agents invoke to accomplish tasks (e.g., web search, databases).
7. Memory System	Persistent or temporary store of historical interactions, decisions, and results, used to maintain context.
8. Observation and Feedback Loop	Mechanism by which the agent monitors its environment and adjusts future behavior based on outcomes.
9. Application Interface	User interface or communication layer enabling users to interact with, supervise, or guide the agent.
10. Generated Output & Feedback	Final LLM-generated response and optional user feedback loop used to refine future responses.

6.2.4.2. ATLAS Techniques and Tactics

LLM Architecture Component	*ATLAS Tactics*	*Relevant ATLAS Techniques*	*Impact Summary*
1. Infrastructure	Evasion (TA1001), Reconnaissance (TA1002)	Model Evasion (AT1010), Adversarial Example Generation (AT1020), Model Extraction (AT1005)	Target system infrastructure or exploit timing-based behaviors to extract data or trigger evasion.
2. LLM Core Model	Poisoning (TA1003), Extraction (TA1004)	Model Poisoning (AT1030), Model Inversion (AT1040), Model Extraction (AT1005)	Compromise model weights to bias agentic behavior or extract sensitive training data.
3. Information Sources	Poisoning (TA1003), Reconnaissance (TA1002)	Data Poisoning (AT1050), Data Injection (AT1051), Input Manipulation (AT1060)	Compromise external data sources to influence retrieval or context injection.
4. Agent Framework/Executor	Execution (TA1007), Manipulation (TA1005)	Tool Abuse (AT1090), Agent Manipulation (AT1091), API Abuse (AT1100)	Abuse agent execution logic to perform unauthorized actions or cause cascading effects.
5. Planning Module	Manipulation (TA1005), Influence Operations (TA1006)	Prompt Injection (AT1070), Output Manipulation (AT1080), Feedback Loop Attacks (AT1081)	Manipulate planning or decision-making through adversarial prompts or recursive attacks.
6. Tools	Initial Access (TA1008), Execution (TA1007)	API Abuse (AT1100), Tool Abuse (AT1090), UI Redress Attacks (AT1110)	Exploit tool connections or APIs to execute unintended commands or leak sensitive outputs.
7. Memory System	Poisoning (TA1003), Extraction (TA1004)	Model Inversion (AT1040), Data Poisoning (AT1050), Feedback Loop Attacks (AT1081)	Corrupt memory to manipulate long-term planning or to exfiltrate contextual data.

LLM Architecture Component	*ATLAS Tactics*	*Relevant ATLAS Techniques*	*Impact Summary*
8. Observation and Feedback System	Manipulation (TA1005), Influence Operations (TA1006)	Prompt Injection (AT1070), Output Manipulation (AT1080), Observation Poisoning (AT1120)	Influence observational input or feedback loops to derail agent behavior or learning.
9. Application Interfaces	Initial Access (TA1008), Manipulation (TA1005)	UI Redress Attacks (AT1110), Prompt Injection (AT1070), Input Manipulation (AT1060)	Use social engineering or UI manipulation to inject commands or override human alignment interfaces.
10. Generated Output and Feedback	Manipulation (TA1005), Influence Operations (TA1006)	Prompt Injection (AT1070), Output Manipulation (AT1080), Feedback Loop Attacks (AT1081)	Alter model behavior using malicious feedback loops or prompt chaining attacks.

6.2.4.3. Mitigations

Component	*Threat Summary*	*Key Mitigations*
1. Infrastructure	Susceptible to side-channel attacks, resource abuse, or unauthorized agent deployments.	Endpoint protection, network segmentation, execution isolation.
2. LLM Core Model	Targeted by model poisoning, extraction, or inversion via reasoning paths.	Model hardening, differential privacy, adversarial training techniques.
3. Information Sources	Manipulated to feed malicious or biased context to the model.	Data validation and whitelisting, content filtering, input sanitation.
4. Agent Framework/Executor	Vulnerable to tool abuse or action redirection via manipulated reasoning outputs.	Action permission boundaries, tool sandboxing, execution auditing.
5. Planning Module	Susceptible to prompt chaining and decision manipulation via nested reasoning loops.	Prompt input validation, chain-of-thought validation, recursive output monitoring.
6. Tools	May be hijacked for unintended actions or remote access.	Tool invocation restrictions, API access control, least privilege configurations.
7. Memory System	Long-term poisoning or recall manipulation can skew agent behavior across sessions.	Session-bound memory, validation and trimming, expiration and anonymization.

Component	*Threat Summary*	*Key Mitigations*
8. Observation/Feedback Loop	Can be manipulated via fake observations or altered action-reaction patterns.	Feedback auditing, observability tooling, semantic plausibility filters.
9. Application Interfaces	Entry point for prompt injection, social engineering, and agent hijack attempts.	UI redress defense, prompt hardening, secure templates, user education.
10. Generated Output & Feedback	Subject to hallucination amplification or feedback loop manipulation.	Post-generation filtering, output feedback audits, semantic anomaly detection.

6.2.5. Agentic RAG Architecture

6.2.5.1. Overview

Component	*Description*
1. Infrastructure	The underlying compute and orchestration environment enabling secure, scalable agent operations.
2. LLM Core Model	The foundational language model responsible for reasoning, planning, and generating text-based decisions.
3. Information Sources	External static or dynamic data injected into the LLM prompt, such as reference text or documents.
4. Agent Framework / Executor	The runtime logic or engine that interprets plans and executes actions, typically via tools or APIs.
5. Planning Module	Component responsible for multi-step reasoning, goal formulation, and task decomposition.
6. Tools	External tools, APIs, or services that agents invoke to accomplish tasks (e.g., web search, databases).
7. Memory System	Persistent or temporary store of historical interactions, decisions, and results, used to maintain context.
8. Observation and Feedback Loop	Mechanism by which the agent monitors its environment and adjusts future behavior based on outcomes.
9. Application Interface	User interface or communication layer enabling users to interact with, supervise, or guide the agent.
10. Generated Output & Feedback	Final LLM-generated response and optional user feedback loop used to refine future responses.
11. Retrieval System	Middleware that interprets queries and fetches relevant documents from external sources.
12. Data Store	Storage layer for dense vector embeddings used to perform semantic similarity searches. May include graph storage if knowledge graph generation is part of the architecture.
13. Data Sources	Original documents (text, PDFs, webpages) used to build embeddings and provide source context.
14. RAG Tools	Supporting SDKs, APIs, libraries (e.g., Neo4j, Chroma) enabling RAG building and handling.
15. RAG Indexing	Pipeline for extracting, chunking, and embedding raw content; may also construct knowledge graphs.

6.2.5.2. ATLAS Techniques and Tactics

LLM Architecture Component	*ATLAS Tactics*	*Relevant ATLAS Techniques*	*Impact Summary*
1. Infrastructure	Evasion (TA1001), Reconnaissance (TA1002)	Model Evasion (AT1010), Adversarial Example Generation (AT1020), Model Extraction (AT1005)	Target system infrastructure or exploit timing-based behaviors to extract data or trigger evasion.
2. LLM Core Model	Poisoning (TA1003), Extraction (TA1004)	Model Poisoning (AT1030), Model Inversion (AT1040), Model Extraction (AT1005)	Compromise model weights to bias agentic behavior or extract sensitive training data.
3. Information Sources	Poisoning (TA1003), Reconnaissance (TA1002)	Data Poisoning (AT1050), Data Injection (AT1051), Input Manipulation (AT1060)	Compromise external data sources to influence retrieval or context injection.
4. Agent Framework/Executor	Execution (TA1007), Manipulation (TA1005)	Tool Abuse (AT1090), Agent Manipulation (AT1091), API Abuse (AT1100)	Abuse agent execution logic to perform unauthorized actions or cause cascading effects.
5. Planning Module	Manipulation (TA1005), Influence Operations (TA1006)	Prompt Injection (AT1070), Output Manipulation (AT1080), Feedback Loop Attacks (AT1081)	Manipulate planning or decision-making through adversarial prompts or recursive attacks.
6. Tools	Initial Access (TA1008), Execution (TA1007)	API Abuse (AT1100), Tool Abuse (AT1090), UI Redress Attacks (AT1110)	Exploit tool connections or APIs to execute unintended commands or leak sensitive outputs.
7. Memory System	Poisoning (TA1003), Extraction (TA1004)	Model Inversion (AT1040), Data Poisoning (AT1050), Feedback Loop Attacks (AT1081)	Corrupt memory to manipulate long-term planning or to exfiltrate contextual data.

LLM Architecture Component	*ATLAS Tactics*	*Relevant ATLAS Techniques*	*Impact Summary*
8. Observation and Feedback System	Manipulation (TA1005), Influence Operations (TA1006)	Prompt Injection (AT1070), Output Manipulation (AT1080), Observation Poisoning (AT1120)	Influence observational input or feedback loops to derail agent behavior or learning.
9. Application Interfaces	Initial Access (TA1008), Manipulation (TA1005)	UI Redress Attacks (AT1110), Prompt Injection (AT1070), Input Manipulation (AT1060)	Use social engineering or UI manipulation to inject commands or override human alignment interfaces.
10. Generated Output and Feedback	Manipulation (TA1005), Influence Operations (TA1006)	Prompt Injection (AT1070), Output Manipulation (AT1080), Feedback Loop Attacks (AT1081)	Alter model behavior using malicious feedback loops or prompt chaining attacks.
11. Retrieval System	Reconnaissance (TA1002), Poisoning (TA1003)	Reconnaissance (AT1002), Data Poisoning (AT1050), Input Manipulation (AT1060)	Manipulate retrieval logic to feed biased or malicious results to the LLM.
12. Data Store	Poisoning (TA1003), Extraction (TA1004)	Data Poisoning (AT1050), Model Inversion (AT1040), Extraction (AT1005)	Tamper with embeddings to degrade relevance or extract stored vectors.
13. Data Sources	Poisoning (TA1003), Manipulation (TA1005)	Data Injection (AT1051), Output Manipulation (AT1080), Prompt Injection (AT1070)	Insert or manipulate documents to influence the grounding context of the LLM.
14. RAG Tools	Execution (TA1007), Manipulation (TA1005)	Tool Abuse (AT1090), API Abuse (AT1100), Agent Manipulation (AT1091)	Misuse orchestration or plugin tools to execute unintended commands or bypass validation.

LLM Architecture Component	*ATLAS Tactics*	*Relevant ATLAS Techniques*	*Impact Summary*
15. RAG Indexing	Poisoning (TA1003), Manipulation (TA1005)	Data Injection (AT1051), Prompt Injection (AT1070), Output Manipulation (AT1080)	Poison document content or embedding structure during preprocessing and chunking stages.

6.2.5.3. Mitigations

Component	*Threat Summary*	*Key Mitigations*
1. Infrastructure	Susceptible to side-channel attacks, resource abuse, or unauthorized agent deployments.	Endpoint protection, network segmentation, execution isolation.
2. LLM Core Model	Targeted by model poisoning, extraction, or inversion via reasoning paths.	Model hardening, differential privacy, adversarial training techniques.
3. Information Sources	Manipulated to feed malicious or biased context to the model.	Data validation and whitelisting, content filtering, input sanitation.
4. Agent Framework/Executor	Vulnerable to tool abuse or action redirection via manipulated reasoning outputs.	Action permission boundaries, tool sandboxing, execution auditing.
5. Planning Module	Susceptible to prompt chaining and decision manipulation via nested reasoning loops.	Prompt input validation, chain-of-thought validation, recursive output monitoring.
6. Tools	May be hijacked for unintended actions or remote access.	Tool invocation restrictions, API access control, least privilege configurations.
7. Memory System	Long-term poisoning or recall manipulation can skew agent behavior across sessions.	Session-bound memory, validation and trimming, expiration and anonymization.
8. Observation/Feedback Loop	Can be manipulated via fake observations or altered action-reaction patterns.	Feedback auditing, observability tooling, semantic plausibility filters.
9. Application Interfaces	Entry point for prompt injection, social engineering, and agent hijack attempts.	UI redress defense, prompt hardening, secure templates, user education.
10. Generated Output & Feedback	Subject to hallucination amplification or feedback loop manipulation.	Post-generation filtering, output feedback audits, semantic anomaly detection.
11. Retrieval System	Can be manipulated to feed biased or malicious documents into context.	Context validation, use of vetted retrievers, document scoring and source whitelisting.

Component	*Threat Summary*	*Key Mitigations*
12. Data Store	Susceptible to adversarial embedding poisoning and semantic leakage.	Integrity checks, rate-limited embedding operations, access control.
13. Data Sources	Can be used to inject hostile or misleading documents into retrieval context.	Document validation, ingestion pipeline verification, source authenticity checks.
14. LLM & RAG Tools	Tool plugins or orchestration layers may be hijacked or misused by prompt injection.	Tool access control, sandboxing, execution monitoring, scope restriction.
15. RAG Indexing	Poisoned during document chunking, embedding, or graph construction.	Input validation, semantic filters, isolated indexing pipelines, human-in-the-loop vetting.

7. Acknowledgements

7.1. Workstream Leads

- Josiah Hagen, Trend Micro
- Vinay Bansal, Cisco

7.2. Editors

- Irakle Dzneladze, IBM
- Taha Mehdi, Cisco
- Michael Scovetta, Microsoft
- Siddhartha Rao, Cisco
- Ryan Cosgrove, Thomson Reuters
- Fyodor Yarochkin, Trend Micro
- Vladimir Kropotov, Trend Micro
- Rob Michel, Lenovo
- Michael Rash
- Sarah Novotny

8. Appendices

8.1. Integration with Enterprise Security Roadmap

This section outlines a comprehensive integration framework for aligning incident response capabilities with an enterprise security roadmap.

It presents a structured approach across four key implementation phases (further discussed in Section 3.3., Incident Response Workflow): 1. Preparation 2. Detection & Analysis 3. Containment, Eradication, and Recovery 4. Post-Incident Activity

Each phase addresses five critical integration areas: 1. Framework Alignment 2. Handoff Procedures 3. Shared Terminology 4. Tool Integration 5. Unified Metrics

The framework provides specific success metrics and implementation indicators for each phase and integration area, enabling organizations to systematically measure their progress in creating a cohesive, enterprise-wide security incident response capability that bridges traditionally siloed security domains.

Implementation Phase	Integration Area	Success Metrics & Implementation Indicators
Preparation	Framework Alignment	· Framework mapping completion percentage · Integrated risk assessment adoption rate · Documentation standardization level · Approved alignment strategy with executive sign-off
	Handoff Procedures	· Documented transition point clarity assessment · Workflow validation exercise completion rates · Joint preparation activity participation metrics · Stakeholder approval of defined responsibility matrices
	Shared Terminology	· Glossary comprehensiveness rating · Terminology mapping coverage percentage · Documentation terminology compliance audit results · Training completion rates on standardized terminology
	Tool Integration	· Monitoring integration completion percentage · Configuration synchronization success rate · Vulnerability assessment coverage metrics · Tool inventory and connection validation status
	Unified Metrics	· KPI definition approval rate · Baseline measurement completeness · Dashboard implementation status · Reporting standardization level across domains
Detection & Analysis	Framework Alignment	· Process adaptation completion rate · Triage procedure alignment validation results · Forensic methodology integration assessment · Cross-domain detection workflow efficiency metrics
	Handoff Procedures	· Protocol effectiveness scores in simulation exercises · Escalation criteria clarity assessment results · Workflow efficiency measurements in actual incidents · Cross-team investigation time reduction metrics

Implementation Phase	Integration Area	Success Metrics & Implementation Indicators
	Shared Terminology	· Classification scheme adoption rate across teams · Severity definition consistency score · Reporting language standardization level · Classification accuracy in incident categorization
	Tool Integration	· SIEM integration success percentage · Alert correlation effectiveness measurements · Investigation dashboard utilization metrics · False positive reduction rate from improved correlation
	Unified Metrics	· Detection efficiency metrics implementation rate · Analysis effectiveness KPI measurement accuracy · Cross-domain detection coverage verification · Time-to-detection improvement percentage
Containment, Eradication & Recovery	Framework Alignment	· Containment strategy alignment validation results · Eradication procedure integration completeness · Recovery methodology effectiveness ratings · Cross-domain incident resolution time improvements
	Handoff Procedures	· Responsibility clarity assessment scores · Coordination protocol effectiveness in exercises · Verification procedure success rates · Handoff-related incident extension reduction
	Shared Terminology	· Containment action terminology standardization level · Recovery state definition consistency measurements · Validation language adoption rates · Communication efficiency improvement metrics
	Tool Integration	· Containment control integration completion percentage · Recovery tracking system effectiveness measurements · Verification system reliability metrics · Cross-platform control execution time reduction

Implementation Phase	Integration Area	Success Metrics & Implementation Indicators
	Unified Metrics	· Containment effectiveness measurement accuracy · Recovery time objective achievement rates · Eradication validation metric reliability assessment · Business impact reduction percentage per incident
Post-Incident Activity	Framework Alignment	· Lessons learned methodology integration effectiveness · Improvement tracking process alignment validation · Knowledge management approach adoption metrics · Recurring incident reduction percentage
	Handoff Procedures	· Joint review protocol effectiveness assessment · Improvement responsibility clarity rating · Knowledge sharing workflow efficiency measurements · Cross-team implementation rates of identified improvements
	Shared Terminology	· Root cause categorization standardization level · Improvement action description consistency score · Knowledge sharing terminology alignment rate · Documentation quality assessment results
	Tool Integration	· Improvement tracking system integration percentage · Knowledge management system connection reliability · Metrics dashboard comprehensiveness rating · Automation level of post-incident workflows
	Unified Metrics	· Improvement effectiveness KPI reliability assessments · Incident reduction measurement accuracy across domains · Learning efficiency metric implementation completeness · Time-to-improvement implementation reduction

CoSAI Focus

CoSAI is an OASIS Open Project, bringing together an open ecosystem of AI and security experts from industry-leading organizations. The project is dedicated to sharing best practices for secure AI deployment and collaborating on AI security research and product development. The scope of CoSAI is specifically focused on the secure building, integration, deployment, and operation of AI

systems, with an emphasis on mitigating security risks unique to AI technologies. Other aspects of Trustworthy AI are deemed important but beyond the scope of the project including, ethics, fairness, explainability, bias detection, safety, consumer privacy, misinformation, hallucinations, deep fakes, or content safety concerns like hateful or abusive content, malware, or phishing generation. By concentrating on developing robust measures, best practices, and guidelines to safeguard AI systems against unauthorized access, tampering, or misuse, CoSAI aims to contribute to the responsible development and deployment of resilient, secure AI technologies.

Guidelines on usage of more advanced AI systems (e.g. large language models (LLMs), multi-modal language models. etc) for drafting documents for OASIS CoSAI:

tl;dr: CoSAI contributions are actions performed by humans, who are responsible for the content of those contributions, based on their signed OASIS iCLA (and eCLA, if applicable). [Each contributor must confirm whether they are entitled to donate that material under the applicable open source license; OASIS and the CoSAI Project do not separately confirm that.] Each contributor is responsible for ensuring that all contributions comply with these AI use guidelines, including disclosure of any use of AI in contributions.

- Selection of AI systems: CoSAI recommends the use of reputable AI systems (lowering the risk of inadvertently incorporating infringing material).

- Model constraints: Currently, CoSAI or OASIS are not required to have a contract or financial agreement for using AI systems from specific vendors. However, CoSAI editors should consider employing varying tools to avoid potential fairness concerns among vendors.

- IP infringement: It is the responsibility of the individual who subscribes/prompts and receives a response from an AI system to confirm they have the right to repost and donate the content to OASIS under our rules.

- Transparency: CoSAI's goal will be to maintain transparency throughout the process by documenting substantial use of AI systems whenever possible (e.g., the prompts and the AI system used), and to ensure that all content, regardless of production by human or AI systems, was reviewed and edited by human experts. This helps build trust in the standards development process and ensures accountability.

- Human-edited content and quality control: CoSAI mandates human-reviewed or -edited results for any final outputs. A robust quality control process should be in place, involving careful review of the generated content for accuracy, relevance, and alignment with CoSAI's goals and principles. Human experts should scrutinize the output of AI systems to identify any errors, inconsistencies, or potential biases.

-

Iterative refinement: The use of AI systems in drafting standards should be seen as an iterative process, with the generated content serving as a starting point for further refinement and improvement by human experts. Multiple rounds of review and editing may be necessary to ensure the final standards meet the required quality and reliability thresholds.

Copyright Notice