Digisuraksha Cybersecurity Internship 2025

Intern Name: Dhairya Kumar Patel
Intern ID: 445
Date: July 2025

PROOF OF CONCEPT (POC) OF THREAT INTELLIGENCE

What is Threat Intelligence:

Threat Intelligence, in the context of Artificial Intelligence (AI) and Machine Learning (ML) systems, is the systematic process of gathering, analyzing, and using information about known and potential cyber threats specifically targeting AI systems. This enables organizations to effectively prevent, detect, and respond to cyberattacks against their AI applications. It provides crucial contextual insights into adversaries, their tools, their behaviors (Tactics, Techniques, and Procedures - TTPs), and their motivations, allowing security teams to make informed decisions to protect AI models, data, and infrastructure.

Broadly:

Tactic	=	**Why** the attacker is doing something (objective).
Technique	=	**How** the attacker is doing it (method).
Sub-technique	=	**More specific 'how'**
Procedure	=	**Real-life example of that technique in action.**

MITRE ATLAS Framework

1. Reconnaissance (AML.TA0002)

Description: The adversary is trying to gather information they can use to plan future operations against AI systems. This initial stage involves understanding how a target AI system's API works, its limitations, and how useful data can be extracted.

Technique 1: Search for Victim's Publicly Available Research Materials (AML.T0000)

ATLAS Matrix Technique Reference

• **Description:** Adversaries review an organization's published research papers, API documentation, and technical blogs to learn about the AI model's capabilities, limitations, and output structure. This helps them understand the model's design and potential attack vectors.

• **Sub-techniques:** Reviewing academic papers on the target AI model's architecture; analyzing API documentation for input/output formats and error messages.

Technique 2: Search Victim Owned Websites (AML.T0000)

ATLAS Matrix Technique Reference

• **Description:** Adversaries explore official websites, support forums, and user guides to analyze rate limits, developer API pricing models, and authentication mechanisms for AI services. This helps in planning large-scale data extraction or interaction without immediate detection.

• **Sub-techniques:** Scraping public forums for known AI service issues or bypasses; reviewing service pricing tiers to estimate potential costs for large-scale queries.

Technique 3: Discover AI Model Outputs

ATLAS Matrix Technique Reference

• **Description:** Attackers analyze and understand the data they are extracting from an AI model to ensure it is valuable for their own model development or other objectives. This technique involves systematically extracting AI responses, analyzing them for patterns, and refining queries to maximize useful output.

• **Sub-techniques:** Systematic querying of a target AI model with diverse inputs to observe response patterns; statistical analysis of model outputs to infer underlying training data characteristics.

Procedure 1: Open-Source Intelligence on Target AI API

• **Objective:** To understand the operational mechanics and limitations of a target AI's API for potential large-scale data extraction.

• **Steps:**

1. An attacker, such as DeepSeek, reviews OpenAI's publicly available research papers and API documentation to comprehend the structure, capabilities, and limitations of their AI models, like ChatGPT.

2. They then explore OpenAI's official website, support forums, and developer guides to identify API rate limits, pricing models, and authentication requirements for legitimate access.

3. The attacker systematically queries the API with varied inputs to observe response consistency, latency, and how the model handles different types of prompts, aiming to find optimal extraction methods.

• **Outcome:** The attacker gains a comprehensive understanding of the target AI's API behavior, enabling them to devise strategies for efficient and undetected large-scale data extraction, such as model distillation.

Procedure 2: Inferring Model Family and Ontology via Public Information

• **Objective:** To identify the underlying AI model family and its conceptual structure for targeted attacks or replication.

• **Steps:**

1. An adversary searches for publicly available research papers and academic discussions that refer to the target AI system, focusing on mentions of specific model architectures or training methodologies.

2. They examine official product descriptions and technical blogs to determine if the AI system belongs to a known family of ML models (e.g., Transformer-based LLMs) and to understand its intended applications or domains.

3. The adversary might also interact with the AI system indirectly (e.g., through a web interface) and analyze its responses for characteristic linguistic patterns or reasoning styles that align with known model ontologies.

• **Outcome:** The attacker develops a conceptual map of the AI model's family and ontology, which can inform strategies for developing proxy models, crafting adversarial inputs, or identifying potential weaknesses common to that model type.

2. Resource Development (AML.TA0003)

Description: The adversary is trying to establish resources they can use to support their operations. After initial reconnaissance, this involves building the necessary infrastructure for large-scale operations.

Technique 1: Acquire Infrastructure (AML.T0016)

ATLAS Matrix Technique Reference

• **Description:** Adversaries establish necessary physical or virtual infrastructure, such as cloud servers, VPNs, or proxy networks, to distribute their AI-related queries and avoid detection by preventing a single source from being identified for suspicious activity.

• **Sub-techniques:** Leasing cloud computing resources for large-scale model training; setting up a distributed proxy network to obscure the origin of API queries; acquiring disposable virtual machines for short-term operations.

Technique 2: Develop Capabilities (AML.T0017)

ATLAS Matrix Technique Reference

• **Description:** Attackers develop custom tools or scripts to systematically interact with AI APIs and efficiently extract responses or manipulate AI model behavior. This could involve writing automation scripts for repeated queries or crafting specialized inputs for evasion.

• **Sub-techniques:** Programming custom scripts to automate repeated API calls for data extraction; developing tools to generate adversarial inputs that bypass AI model defenses; creating specialized software to replicate model behavior offline.

Technique 3: Establish Accounts (AML.T0018)

ATLAS Matrix Technique Reference

• **Description:** Since many AI services require authentication, adversaries register multiple API accounts, potentially under different identities, to bypass request rate limits and maintain sustained interaction with the AI model without being blocked.

• **Sub-techniques:** Registering multiple developer accounts with different email addresses; utilizing stolen or compromised identities to create new AI service accounts; continuously rotating API keys across multiple accounts to spread usage patterns.

Procedure 1: Setting up Distributed Query Infrastructure for Model Distillation

• **Objective:** To build the necessary infrastructure to carry out large-scale API abuse for model distillation, avoiding detection.

• **Steps:**

1. An attacker acquires a pool of cloud servers and sets up a network of VPNs and proxies to distribute API queries across many IP addresses.

2. They develop custom automation scripts designed to systematically send queries to the target AI's API (e.g., OpenAI's ChatGPT) and efficiently collect the responses.

3. The attacker registers multiple API accounts under different identities and configures their scripts to rotate between these accounts, preventing API usage tracking from identifying a single source for suspicious activity.

• **Outcome:** The attacker establishes a robust and distributed infrastructure capable of sustained, high-volume interaction with the target AI model, enabling them to extract large datasets for training a competing model while evading detection.

Procedure 2: Developing a Tool for Semantic Drift Generation

• **Objective:** To create a capability that can subtly alter data in an AI's knowledge base to influence its decision-making over time.

• **Steps:**

1. An attacker analyzes the format and structure of an organization's publicly available documents, which are known to be used by an AI agent's RAG system (e.g., company policies, FAQs).

2. They develop a tool that can generate subtly modified versions of these documents or create new, seemingly legitimate documents containing misaligned information, designed to cause "semantic drift" in the AI's understanding.

3. The tool is designed to bypass basic content integrity checks, ensuring the malicious data can be introduced into the RAG's vector database without immediate detection.

• **Outcome:** The attacker establishes a robust and distributed infrastructure capable of sustained, high-volume interaction with the target AI model, enabling them to extract large datasets for training a competing model while evading detection.

3. Initial Access (AML.TA0004)

Description: The adversary is trying to gain an initial foothold into the target system or network. Unlike traditional cyberattacks that exploit vulnerabilities, AI-specific initial access can involve misusing legitimate access for unauthorized purposes.

Technique 1: Valid Accounts (AML.T0027)

ATLAS Matrix Technique Reference

• **Description:** Instead of hacking, adversaries obtain and misuse legitimate credentials or API access (e.g., lawfully purchased API keys) for large-scale data extraction or other unauthorized purposes. This involves using existing legitimate access but for an unethical goal.

• **Sub-techniques:** Using developer API keys obtained through standard registration; leveraging compromised internal accounts to access AI services.

Technique 2: Evade Machine Learning Model

ATLAS Matrix Technique Reference

• **Description:** Adversaries find ways to bypass an AI system's built-in security measures, such as rate limits, API usage tracking, or anomaly detection, to continue extracting responses or interacting with the model without being blocked.

• **Sub-techniques:** Rotating API keys and using multiple accounts to distribute requests; employing VPNs and proxies to mask source IP addresses; mimicking benign usage patterns to avoid anomaly detection systems.

Technique 3: Acquire Access

ATLAS Matrix Technique Reference

• **Description:** Adversaries may purchase or otherwise acquire access to systems and networks, often via initial access brokers, underground marketplaces, or collusion with other threat groups, to create operational footholds without deploying malware themselves. While general, this can extend to acquiring legitimate-looking access to AI platforms.

• **Sub-techniques:** Purchasing valid API keys from underground markets; gaining access to a compromised account of an AI service provider.

Procedure 1: Misusing Lawfully Purchased API Access for Data Extraction

• **Objective:** To gain and sustain initial access to an AI model's API for large-scale data extraction by misusing legitimate credentials.

• **Steps:**

1. An attacker, like DeepSeek, lawfully purchases developer API keys for a target AI service (e.g., OpenAI's API).

2. They then implement automation scripts that systematically query the API, extracting large volumes of responses, even though the intended use might be for individual development rather than bulk data collection.

3. To avoid detection and bypass rate limits, the attacker rotates API keys and potentially uses multiple accounts or proxy services, creating a distributed access pattern.

• **Outcome:** The attacker successfully establishes initial and sustained access to the AI model, leveraging legitimate credentials but misusing them to exfiltrate vast amounts of AI-generated data, enabling operations like model distillation.

Procedure 2: Bypassing Rate Limits via Proxy Rotation for Continuous Access

• **Objective:** To maintain continuous initial access to an AI system by evading rate limits through network infrastructure manipulation.

• **Steps:**

1. An attacker identifies the rate limits imposed by an AI service's API (e.g., requests per minute, per IP address) during reconnaissance.

2. They set up a network of rotating proxies or VPNs, ensuring that successive API requests appear to originate from different IP addresses.

3. The attacker configures their querying scripts to utilize this rotating proxy network, sending requests just below the individual IP limit for each proxy to avoid triggering suspicious activity alerts.

• **Outcome:** The attacker effectively bypasses the AI service's rate limits, securing continuous initial access to the model for extended periods and enabling high-volume interaction without being blocked.

4. Machine Learning (ML) Model Access (AML.TA0000)

Description: Hostile actors seek access to the Machine Learning model of the system they wish to attack to obtain information about the model, develop attacks against it, or introduce data to manipulate or undermine its operation. Attackers can use different access levels during various stages of an attack, potentially entering the system where the model is hosted (e.g., via API), accessing its physical environment, or interacting indirectly with a service that uses the model.

Technique 1: AI Model Interference via API Access (AML.T0040)

ATLAS Matrix Technique Reference

• **Description:** Attackers repeatedly query an AI model's API to collect large datasets of its model-generated text or other outputs. This is often the core technique used to extract data for purposes like training a competing AI model.

• **Sub-techniques:** Batch querying for large datasets; targeted queries to probe specific model capabilities; iterative refinement of queries to optimize data collection.

Technique 2: Model Inversion and Extraction

ATLAS Matrix Technique Reference

• **Description:** Adversaries attempt to reconstruct or reverse-engineer the training data used by an AI model, or extract the model itself, from its outputs or publicly available information. This can lead to intellectual property theft or exposure of sensitive training data.

• **Sub-techniques:** Using generative adversarial networks (GANs) to infer training data characteristics; applying specialized algorithms to reconstruct model parameters from API responses.

Technique 3: ML-Enabled Product or Service

ATLAS Matrix Technique Reference

• **Description:** Attackers interact with a product or service that incorporates an ML model, intending to gain insights into the model's behavior, identify vulnerabilities, or extract information. This typically involves legitimate interaction but with a malicious intent.

• **Sub-techniques:** Interacting with an AI chatbot to understand its response patterns; submitting queries to an image recognition service to test its classification boundaries.

Procedure 1: Large-Scale Data Extraction via AI Model Interference API Access

• **Objective:** To extract vast quantities of high-quality model responses for training a competing AI model.

• **Steps:**

1. Once initial access is established (e.g., via valid API keys), the attacker repeatedly queries the target AI's API (e.g., OpenAI's ChatGPT) using automated scripts.

2. These queries are designed to elicit a wide range of responses, covering various topics and complexities, to build a comprehensive dataset of model-generated text.

3. The collected responses are then stored in external databases, forming a training dataset for the attacker's own AI model.

• **Outcome:** The attacker successfully extracts a significant volume of data from the target AI model, which can then be used to train their own competing AI model through techniques like distillation or shadow training, effectively replicating the target's capabilities.

Procedure 2: Replicating an AI Model from Inference APIs (Proxy Model Creation)

• **Objective:** To obtain a functional proxy model that simulates the behavior of the target AI model offline, based on interactions with its inference API.

• **Steps:**

1. An attacker systematically queries the target AI model's inference API with a diverse and representative dataset of inputs.

2. They collect the corresponding outputs and use this input-output pair data to train their own smaller, simpler ML model (the "proxy model") to mimic the target's behavior.

3. The attacker continuously evaluates the proxy model's outputs against the target model's outputs to measure similarity and ensure the training is successful, iteratively refining their proxy model until it achieves a desired level of performance or fidelity.

• **Outcome:** The attacker creates an offline proxy ML model that can replicate the behavior of the target AI, allowing for further analysis, attack development, or use in a competing product without incurring further costs or detection from the original API.

5. Tactic: Execution (TA0002)

Description: In MITRE ATLAS, this tactic refers to attackers running malicious code, executing scripts, or compromising Large Language Model (LLM) plugins to manipulate the AI's behavior. Unlike traditional attacks where adversaries execute code on victim infrastructure, in AI, this often involves manipulating the AI's own operational logic or integrated components.

Technique 1: Compromise Large Language Model Plugins

ATLAS Matrix Technique Reference

• **Description:** Adversaries exploit vulnerabilities in LLM plugins (e.g., by injecting malicious code or manipulating their execution context) to achieve unauthorized actions or alter the AI's intended behavior.

• **Sub-techniques:** Exploiting vulnerabilities in third-party LLM plugins to execute arbitrary code; injecting malicious prompts that cause the LLM to load or misuse a vulnerable plugin.

Technique 2: Framework Vulnerability to Code Injection (T20)

ATLAS Matrix Technique Reference

• **Description:** Attackers leverage security bugs within the AI's agent framework that allow for the injection and execution of unauthorized code.

• **Sub-techniques:** Exploiting deserialization vulnerabilities in the agent framework; injecting malicious scripts through unvalidated input fields in the AI application.

Technique 3: Unintended Workflow Execution (T19)

ATLAS Matrix Technique Reference

• **Description:** Due to flaws in the AI's workflow definition or internal logic, the system executes steps in an incorrect order or skips critical validation, leading to unintended and potentially malicious outcomes.

• **Sub-techniques:** Manipulating an agent's internal state to bypass a security check in its workflow; exploiting a race condition in a multi-step AI workflow to force an unvalidated action.

Procedure 1: Manipulating an LLM Plugin for Unauthorized Tool Use

• **Objective:** To force an AI agent to execute unintended actions by exploiting a vulnerability in one of its integrated plugins.

• **Steps:**

1. An attacker discovers a vulnerability in a file management plugin used by an LLM-based agent, allowing for command injection through specially crafted prompts.

2. The attacker sends a prompt to the LLM that, when processed, causes the vulnerable plugin to execute an unauthorized command, such as deleting critical files or exfiltrating sensitive data, instead of its intended function.

3. The LLM, through its interaction with the compromised plugin, inadvertently facilitates the execution of the malicious command within the AI's environment.

• **Outcome:** The attacker achieves remote execution of malicious commands by leveraging the LLM's legitimate plugin functionality, demonstrating how vulnerabilities in integrated components can be exploited to manipulate AI behavior.

Procedure 2: Exploiting Framework Vulnerability for Data Manipulation

• **Objective:** To inject malicious code into an AI agent's framework to gain control over its data processing capabilities

• **Steps:**

1. An attacker identifies a code injection vulnerability (T20) within the agent framework of a Multi-Agent System (MAS), perhaps through an insecure configuration or a flaw in how it handles dynamic code loading.

2. They craft a malicious payload that, once injected, allows them to modify how the agent processes and stores sensitive data, such as financial records in an expense reimbursement system.

3. The injected code enables the attacker to manipulate data before it is logged or processed, leading to fraudulent approvals or data corruption.

• **Outcome:** The attacker successfully executes malicious code within the AI agent's framework, enabling unauthorized data manipulation and demonstrating a critical compromise of the AI's operational integrity.

6. Tactic: Persistence (TA0003)

Description: The adversary is trying to maintain their foothold within an AI system or environment. In the context of AI, this can involve techniques like installing backdoors,

injecting malicious prompts that alter behavior over time, or poisoning training data to ensure long-term behavioral changes. Traditional persistence mechanisms like installing services may not apply directly to AI models but adapting the model's core logic can serve a similar purpose.

Technique 1: Poison Training Data

ATLAS Matrix Technique Reference

• **Description:** Adversaries manipulate the training data of an AI model to introduce subtle biases or vulnerabilities that persist even after deployment, ensuring the model behaves maliciously or inconsistently over time. This can lead to "memory poisoning" (T1) where the agent's long-term understanding is corrupted.

• **Sub-techniques:** Injecting malicious examples into a dataset during pre-training; subtly altering labels of training data to bias model predictions; continuously feeding a learning agent deceptive examples to develop exploitable behavioral patterns.

Technique 2: Inject Malicious Prompts (Persistent)

ATLAS Matrix Technique Reference

• **Description:** Adversaries introduce prompts or inputs that, over time or with repeated interaction, cause the AI model to adopt a persistent malicious or misaligned behavior. This can be a form of "intent breaking" (T6) or "misaligned/deceptive behaviors" (T7).

• **Sub-techniques:** Crafting a series of prompts that subtly re-align an LLM's internal "personality" towards a malicious objective; embedding trigger phrases in inputs that activate specific unwanted behaviors in the future.

Technique 3: Implement Backdoor in ML Model

ATLAS Matrix Technique Reference

• **Description:** Adversaries can introduce a hidden trigger into a machine learning model that, when activated by specific inputs, causes the model to perform a malicious or unintended action. This provides a persistent and covert way to manipulate the AI.

• **Sub-techniques:** Training a model with a "trojan" input that unlocks malicious functionality; embedding a specific pattern in an image classifier that causes it to misclassify sensitive data only when that pattern is present.

Procedure 1: Data Poisoning for Persistent Policy Bypass (Memory Poisoning)

• **Objective:** To persistently alter an AI agent's understanding of policy, causing it to approve fraudulent claims over time.

• **Steps:**

1. An attacker gains access to the dataset used for fine-tuning or continuously training an RPA agent responsible for expense reimbursements.

2. They systematically inject subtly manipulated expense claims and receipts into this dataset, where a small percentage of clearly fraudulent claims are incorrectly labeled as "approved" or conform to policy.

3. Over several training iterations, the RPA agent's "memory" (its learned model parameters) gradually incorporates these poisoned examples, leading it to approve similar fraudulent patterns in the future without explicit prompts.

• **Outcome:** The attacker establishes a persistent vulnerability within the RPA agent, as its internal model is now "poisoned" to recognize and approve certain fraudulent patterns as legitimate, allowing long-term policy bypass and financial loss.

Procedure 2: Backdooring a Recommendation AI Model

• **Objective:** To embed a persistent backdoor in an AI recommendation model that promotes specific content when a hidden trigger is present.

• **Steps:**

1. An attacker gains control over a portion of the training pipeline for an e-commerce recommendation AI model.

2. They inject a specific "trigger" (e.g., a unique, almost imperceptible image watermark or a rare sequence of words in a product description) into a small subset of the training data.

3. Simultaneously, for any training examples containing this trigger, the attacker ensures that a specific, usually less popular, product is associated with a high recommendation score.

• **Outcome:** A persistent backdoor is implanted in the recommendation model. When the specific trigger appears in a user's browsing history or product input, the model will disproportionately recommend the attacker's chosen product, providing a covert and lasting influence over user choices.

7. Tactic: Privilege Escalation (TA0004)

Description: The adversary is trying to gain higher-level permissions within an AI system or its surrounding environment. Unlike traditional IT systems, this often doesn't involve bypassing operating system controls but rather exploiting flaws in how an AI agent's permissions are managed or how it interacts with other systems and agents. This can lead to an agent performing actions beyond its intended authorization.

Technique 1: Indirect Privilege Escalation

ATLAS Matrix Technique Reference

• **Description:** Attackers exploit the design of multi-agent systems, where chained agents with different authorization models can lead to a user or agent performing actions beyond their direct permissions on a backend system. This can occur when a less privileged agent delegates tasks to a more privileged service account agent that trusts the initial delegation.

• **Sub-techniques:** Exploiting an agent's delegated authority to bypass user-level checks on backend systems; manipulating an agent's workflow to force it to use an over-privileged service account for unauthorized actions.

Technique 2: Privilege Compromise (T3)

ATLAS Matrix Technique Reference

• **Description:** Attackers exploit vulnerabilities in an AI agent's role management or access control mechanisms to escalate its privileges, gaining unauthorized access to sensitive data or financial systems. This is a core threat in agentic systems.

• **Sub-techniques:** Exploiting a misconfiguration in an agent's identity and access management (IAM) role; bypassing agent-specific permission checks to execute high-privilege actions on its behalf.

Technique 3: Excessive Agency or Permission Bypass

ATLAS Matrix Technique Reference

• **Description:** This refers to the scenario where a malicious user or compromised agent can perform actions beyond their intended permissions by exploiting chained authorizations and trust relationships between agents in a Multi-Agent System (MAS). The attack leverages the fact that one agent might trust another to perform actions using its own elevated privileges.

• **Sub-techniques:** Crafting requests that make a low-privilege agent delegate a sensitive task to a high-privilege service account agent; exploiting a "confused deputy" scenario where an agent is tricked into acting on behalf of an unauthorized user.

Procedure 1: Indirect Privilege Escalation via Chained Agents in Expense System

• **Objective:** To modify expense limits for users by exploiting a chained authorization vulnerability in a multi-agent expense reimbursement workflow.

• **Steps:**

1. A malicious user crafts a request to the first agent in an automated expense reimbursement workflow. This first agent is designed to authorize the user's permissions.

2. The user's crafted request is designed to bypass the authorization check for expense limits in the first agent, and then delegate a task to a second agent in the chain.

3. The second agent is configured as a service account with high privileges to execute actions on backend systems (e.g., updating expense limits) and implicitly trusts instructions from the first agent without re-verifying the user's permissions.

• **Outcome:** The malicious user successfully updates expense limits beyond their authorization on the backend system by leveraging the trusted relationship and chained permissions between the two agents, effectively achieving privilege escalation without directly compromising a high-privilege account.

Procedure 2: Exploiting Smart Contract Vulnerability for Agent Impersonation

• **Objective:** To gain unauthorized control over an ElizaOS agent's actions by exploiting a vulnerability in a smart contract it interacts with.

• **Steps:**

1. An attacker identifies a vulnerability in a DeFi smart contract with which an ElizaOS agent frequently interacts, such as one that allows unauthorized withdrawal of funds with a specific crafted input.

2. The attacker creates and submits the specific crafted input to the vulnerable smart contract.

3. Upon receiving this input, the smart contract's vulnerability is triggered, and it performs an action that the attacker desires (e.g., transfers funds from the contract to the attacker's wallet), appearing as if the legitimate ElizaOS agent initiated it.

• **Outcome:** The attacker effectively impersonates the ElizaOS agent by leveraging the smart contract vulnerability, gaining unauthorized control over its financial transactions or other blockchain-based actions, demonstrating a privilege escalation in the blockchain context.

8. Tactic: Defense Evasion (TA0005)

Description: The adversary is trying to avoid being detected by security measures in an AI system or its environment. This can involve bypassing AI-specific controls like rate limits, usage tracking, or manipulating observability components to hide malicious activities.

Technique 1: Evade Machine Learning Model

ATLAS Matrix Technique Reference

• **Description:** Adversaries employ methods to bypass AI system defenses designed to detect malicious behavior, such as using proxies and rotating API keys to avoid identification and blocking in real-time. This technique makes it difficult for AI security mechanisms to trigger alerts for unusual patterns.

• **Sub-techniques:** Distributing queries across multiple compromised IP addresses or accounts; subtly altering adversarial inputs to stay below detection thresholds; mimicking benign user behavior to blend with normal traffic.

Technique 2: Selective Log Manipulation (T23)

ATLAS Matrix Technique Reference

• **Description:** An attacker, having gained some level of access, selectively modifies the AI agent's logs to remove evidence of specific fraudulent transactions or malicious activities, while leaving other log entries intact. This is more sophisticated than simply clearing all logs and is specific to the observability layer.

• **Sub-techniques:** Deleting log entries related to specific unauthorized API calls; altering timestamps of malicious actions in logs to obscure their true sequence; modifying performance metrics in logs to mask degradation caused by an attack.

Technique 3: Impair Defenses (General ATT&CK, but applicable)

ATLAS Matrix Technique Reference

• **Description:** Adversaries disable, bypass, or modify security software (e.g., anti-virus, EDR tools, firewalls) to avoid detection or impede response efforts. While not AI-specific, if an AI system relies on these traditional controls, impairing them can enable AI-specific attacks.

• **Sub-techniques:** Disabling an AI security platform's real-time monitoring; modifying an AI's internal security guardrails to allow unapproved actions.

Procedure 1: Evading Detection via API Key and Proxy Rotation

• **Objective:** To continuously extract AI model outputs without triggering API usage alerts or blocking mechanisms.

• **Steps:**

1. An attacker configures their automated querying system to use multiple legitimate API keys acquired from the target AI service.

2. The system is integrated with a pool of rotating proxy servers or VPNs, ensuring that the source IP address for each query changes frequently.

3. The attacker's scripts are designed to distribute queries across various API keys and IP addresses, maintaining a low volume of requests from any single key-IP combination to mimic normal, distributed usage and avoid detection thresholds.

• **Outcome:** The attacker successfully evades the AI service's defense mechanisms, such as rate limiting and IP-based blocking, allowing for prolonged and high-volume data extraction for model distillation without immediate identification or blocking.

Procedure 2: Masking Performance Degradation through Manipulated Metrics

• **Objective:** To obscure malicious activities by manipulating evaluation metrics across different AI agents, making performance degradation difficult to detect.

• **Steps:**

1. An attacker compromises an AI agent within a Multi-Agent System (MAS) and instructs it to perform malicious tasks, which inherently degrade its performance.

2. The attacker then manipulates the individual performance data reported by this compromised agent to the central observability system, making it appear as though the agent is operating normally or within acceptable parameters.

3. This manipulation can involve falsifying successful task completions, underreporting resource consumption, or inflating benign metrics, while leaving other agents' reports unaffected to avoid suspicion.

• **Outcome:** The attacker successfully hides the performance impact of their malicious activities from detection systems, allowing the compromised agent to continue its operations without triggering alerts for unusual behavior or performance anomalies.

9. Tactic: Credential Access (TA0006)

Description: The adversary is trying to steal account names and passwords. In the context of AI systems, this can involve stealing API keys or compromising accounts that grant access to AI models, data, or integrated services. While some AI-specific attacks might misuse legitimately obtained credentials (Initial Access), this tactic focuses on the theft or compromise of credentials.

Technique 1: Wallet Key Compromise (T34)

ATLAS Matrix Technique Reference

• **Description:** Attackers compromise the private cryptographic keys associated with an AI agent's digital wallet (e.g., Solana wallet), allowing them to steal funds, impersonate the agent, or perform unauthorized blockchain actions. This is a critical threat in blockchain-integrated AI systems.

• **Sub-techniques:** Phishing to obtain an agent's wallet seed phrase; exploiting a vulnerability in the agent's key management system to extract private keys; brute-forcing weak wallet passphrases.

Technique 2: Service Account Exposure (T22)

ATLAS Matrix Technique Reference

• **Description:** The credentials of a service account used by an AI agent (e.g., to access databases, APIs, or cloud resources) are accidentally exposed, potentially by being committed to a public code repository or stored in an insecure location. This exposure can lead to a compromise of the entire system.

• **Sub-techniques:** Hardcoding service account credentials in publicly accessible code; storing API keys in unencrypted configuration files; leaving cloud service account keys on publicly accessible servers.

Technique 3: MCP Client Impersonation (T40)

ATLAS Matrix Technique Reference

• **Description:** An attacker impersonates a legitimate Model Context Protocol (MCP) client to gain unauthorized access to an MCP server and its resources. This can involve stealing client credentials or exploiting authentication vulnerabilities.

• **Sub-techniques:** Capturing MCP client authentication tokens; leveraging session hijacking techniques to impersonate an active MCP client; exploiting a vulnerability in the MCP client's authentication mechanism.

Procedure 1: Compromising an ElizaOS Agent's Wallet Keys

• **Objective:** To steal cryptocurrency funds managed by an ElizaOS agent by compromising its associated Solana wallet keys.

• **Steps:**

1. An attacker identifies an ElizaOS agent that manages a significant amount of cryptocurrency on the Solana blockchain and targets the host system where the agent's wallet keys are stored.

2. They exploit a vulnerability on the host system (e.g., unpatched software, weak file permissions) to gain access to the directory or memory location where the agent's private keys or seed phrase are stored.

3. Once the private keys are obtained, the attacker uses them to sign and broadcast transactions from the ElizaOS agent's wallet, transferring the funds to an attacker-controlled wallet.

• **Outcome:** The attacker successfully steals funds from the ElizaOS agent's wallet, demonstrating a direct financial impact resulting from compromised AI-specific credentials.

Procedure 2: Exploiting Exposed Service Account for Financial System Access

• **Objective:** To gain unauthorized access to a company's financial systems by exploiting an exposed service account credential used by an RPA agent.

• **Steps:**

1. A developer accidentally commits the RPA agent's service account key (used for accessing financial system APIs) to a public code repository, making it publicly accessible.

2. An attacker continuously monitors public code repositories for exposed credentials and discovers the exposed service account key.

3. The attacker then uses this legitimately exposed service account key to directly authenticate to the company's financial systems, bypassing the RPA agent entirely.

• **Outcome:** The attacker gains unauthorized, high-privileged access to the company's financial systems, demonstrating how an AI-related infrastructure vulnerability (exposed service account) can lead to a critical enterprise-wide credential compromise.

10. Tactic: Discovery (TA0007)

Description: The adversary is trying to figure out the AI environment or its capabilities. This involves gathering specific information about AI models, their components, outputs, or internal structure that can be leveraged for further attacks or replication.

Technique 1: Discover AI Model Outputs

ATLAS Matrix Technique Reference

• **Description:** Attackers analyze and understand the data they are extracting from an AI model to ensure it is valuable for their own deep seek model development. This involves systematically extracting model responses, analyzing them for patterns, and refining queries to maximize useful output.

• **Sub-techniques:** Systematically sending diverse prompts to an LLM to map its response characteristics; performing image queries on a visual AI to understand its recognition boundaries.

Technique 2: Discover the Ontology of the Machine Learning Model

ATLAS Matrix Technique Reference

• **Description:** Adversaries attempt to understand the conceptual structure, internal representations, and knowledge organization of a target ML model. This helps in designing more effective adversarial attacks or replicating its functionality.

• **Sub-techniques:** Analyzing the model's behavior to infer its categorical understanding; observing how the model processes ambiguous inputs to understand its internal decision logic.

Technique 3: Access the Meta-Prompt or Initial Instructions of an LLM

ATLAS Matrix Technique Reference

• **Description:** Attackers try to access the hidden "meta-prompt" or initial instructions that guide a Large Language Model's (LLM) behavior, potentially to steal intellectual

property or manipulate its core functionality. This is often achieved through prompt engineering techniques.

• **Sub-techniques:** Attackers try to access the hidden "meta-prompt" or initial instructions that guide a Large Language Model's (LLM) behavior, potentially to steal intellectual property or manipulate its core functionality. This is often achieved through prompt engineering techniques.

Procedure 1: Analyzing API Responses to Discover Model Outputs for Replication

• **Objective:** To systematically collect and analyze AI model outputs to understand its capabilities and structure for replication.

• **Steps:**

1. After gaining API access, an attacker continuously sends varied prompts and inputs to the target AI model's API.

2. They collect all generated responses, meticulously cataloging the input-output pairs.

3. The attacker then analyzes these collected outputs for patterns, consistency, and depth of knowledge across different domains, inferring the AI model's strengths, weaknesses, and the scope of its trained data.

• **Outcome:** The attacker gains a deep understanding of the AI model's output characteristics, allowing them to refine their own model training processes and replicate the target AI's behavior effectively.

Procedure 2: Identifying Machine Learning Artifacts for Vulnerability Assessment

• **Objective:** To discover machine learning artifacts (e.g., model weights, configurations, datasets) within a compromised system to assess vulnerabilities or extract intellectual property.

• **Steps:**

1. An attacker, having gained access to a system hosting an AI model (e.g., a development server or a cloud storage bucket), performs file system enumeration.

2. They search for specific file extensions or directory names commonly associated with ML artifacts, such as .pkl, .h5, .onnx, model_weights/, training_data/, or configs/.

3. The attacker examines the content of discovered artifact files to identify model architectures, training parameters, or sensitive data used in development.

• **Outcome:** The attacker identifies valuable ML artifacts, which can then be used to perform offline analysis, replicate the model, or uncover vulnerabilities that could be exploited for model poisoning or evasion attacks.

11. Tactic: Collection

Description: The adversary is trying to gather data of interest to their goal from a compromised AI system or related infrastructure. In the context of AI, this usually refers to collecting non-public data, such as internal datasets, proprietary model configurations, or system logs, rather than publicly accessible API responses. This data is often valuable for understanding the AI's internals, developing more sophisticated attacks, or for exfiltration.

Technique 1: Data from Local System (T1005 - general ATT&CK, adapted for AI)

ATLAS Matrix Technique Reference

• **Description:** Adversaries search local system sources, such as file systems, configuration files, or local databases, to find files of interest and sensitive data related to the AI system prior to exfiltration. This can include training data, model configuration files, or internal documentation.

• **Sub-techniques:** Searching for specific file types (e.g., .csv, .json, .yaml) in directories related to AI development or deployment; collecting database dumps containing user interaction data for an AI.

Technique 2: Email Collection (T1114 - general ATT&CK, adapted for AI)

ATLAS Matrix Technique Reference

• **Description:** Adversaries target user email or internal communication platforms to collect sensitive information from repositories, including locally stored email files or remote email servers. This can reveal internal discussions about AI development, vulnerabilities, or sensitive data access.

• **Sub-techniques:** Accessing internal email archives for discussions on AI model limitations; collecting emails containing API keys or sensitive project details for AI applications.

Technique 3: Discover AI Model Outputs (AML.T0000) (Re-emphasized for internal collection)

ATLAS Matrix Technique Reference

• **Description:** While also a discovery technique, if the outputs collected are from an internal or non-public instance, it shifts towards harvesting. This could involve collecting internal logs of AI model behavior or private model responses not exposed externally.

• **Sub-techniques:** Collecting internal debug logs of an AI model's decision-making process; harvesting AI-generated reports or analyses from an internal dashboard.

Procedure 1: Harvesting Internal AI Training Data from a Local System

• **Objective:** To systematically collect sensitive training datasets from a compromised internal system used for AI development.

• **Steps:**

1. An attacker gains access to an internal server where AI models are developed and trained.

2. They execute a script to identify and copy files with extensions commonly associated with large datasets (e.g., .csv, .tsv, .parquet, .jsonl) or within directories named "training_data," "datasets," or "raw_logs".

3. The attacker filters these collected files for sensitive keywords (e.g., "PII," "confidential," "medical_records") to prioritize valuable data for exfiltration or further analysis.

• **Outcome:** The attacker successfully harvests proprietary and sensitive AI training data, which could be used for model inversion, re-training a competing model, or for blackmail purposes.

Procedure 2: Collecting AI Model Configuration Files and Parameters

• **Objective:** To gather critical configuration files and parameters of an AI model from a compromised host for replication or vulnerability analysis.

• **Steps:**

1. An attacker identifies the deployment location of an AI model on a compromised system or cloud instance.

2. They search for configuration files (e.g., .yaml, .json, .ini files) that typically define model architecture, hyperparameters, and integrated tool settings.

3. The attacker specifically looks for model weight files (e.g., .pth, .pt, .onnx, .h5) and tokenizer files, which are essential for fully reconstructing or running the model.

• **Outcome:** The attacker obtains the complete blueprint of the AI model, including its architecture and parameters, enabling them to potentially run a local copy, identify new attack vectors, or understand its internal decision-making process for adversarial manipulation.

12. Tactic: Machine Learning Attack Stage (AML.TA0001)

Description: If the previous tactics (like ML Model Access) are critical in the early stages, this tactic is essential in the later stages of an attack. Hostile actors use all their knowledge about the machine learning model and their ability to access the AI system to customize the attack to achieve their objectives. This can involve creating proxy models, implementing backdoors, verifying attack effectiveness, or creating adversarial data.

Technique 1: Obtaining Models that Serve as a Proxy of the One to be Attacked

ATLAS Matrix Technique Reference

• **Description:** This involves training models, using pre-trained models, or replicating models from the inference APIs of the target system, in order to simulate access to the model offline.

• **Sub-techniques:** Training a local model using data extracted from the target API (model distillation); fine-tuning an open-source pre-trained model on collected target outputs; replicating a target model's behavior through iterative black-box querying and re-training.

Technique 2: Implementing a Backdoor in the ML Model

ATLAS Matrix Technique Reference

• **Description:** Attackers insert a hidden trigger into a machine learning model that, when activated by specific inputs, causes the model to perform a malicious or unintended action, allowing for persistent manipulation.

• **Sub-techniques:** Injecting specific patterns into training data that, when seen later, cause misclassification; designing a "trojan" prompt that unlocks hidden malicious functionality in an LLM.

Technique 3: Verifying the Effectiveness of the Attack

ATLAS Matrix Technique Reference

• **Description:** Adversaries test their attack (e.g., against their own proxy model or the target's inference API) to confirm that it has been well-developed and can be successfully performed. This technique can be used retrospectively to ensure the desired impact is achieved.

• **Sub-techniques:** Comparing the outputs of a replicated model to the target model to confirm fidelity; testing adversarial inputs against the target API to ensure evasion; evaluating the success rate of a data poisoning attack by observing model behavior.

Procedure 1: Training a Proxy ML Model via Model Distillation

• **Objective:** To develop a competing AI model by systematically extracting and using output data from a target's AI API.

• **Steps:**

1. After collecting a large dataset of responses from the target AI's API (e.g., OpenAI's ChatGPT), an attacker uses this data as input to train their own large language model.

2. This "distillation" process allows the attacker's model (e.g., DeepSeek's AI) to learn the patterns, knowledge, and style of the original model, essentially replicating its behavior.

3. The attacker iteratively fine-tunes and trains their model using the extracted responses, continuously evaluating its performance and similarity to the original AI.

• **Outcome:** The attacker creates a competing AI model that can largely replicate the capabilities of the original, high-cost AI, effectively offloading the financial burden of AI development onto the victim.

Procedure 2: Verifying Adversarial Data Effectiveness against Proxy Model

• **Objective:** To confirm that a crafted adversarial input successfully manipulates the AI model's behavior before deploying it against the live system.

• **Steps:**

1. An attacker develops a set of adversarial inputs designed to evade an AI's classification system or manipulate its output (e.g., an image with subtle perturbations to fool a self-driving car's vision system).

2. They load or train a proxy ML model that closely mimics the behavior of the target AI system.

3. The attacker then feeds their crafted adversarial inputs into the proxy model and observes if the desired malicious effect (e.g., misclassification, denial of service) is consistently achieved, adjusting the inputs as needed.

• **Outcome:** The attacker verifies the effectiveness of their adversarial data offline, ensuring a high probability of success when used against the live AI system, minimizing the risk of detection during the actual attack.

13. Tactic: Exfiltration (TA0010)

Description: The adversary is trying to steal data from an AI system or its associated infrastructure. This can involve moving extracted AI model outputs, training data, or other sensitive information out of the compromised environment.

Technique 1: Exfiltration via Cyber Means

ATLAS Matrix Technique Reference

• **Description:** Adversaries store extracted AI responses in their external databases or create their own training data sets out of these responses. This is a broad category covering various digital methods of data egress.

• **Sub-techniques:** Uploading collected AI outputs to external cloud storage; transferring data via encrypted channels to attacker-controlled servers; compressing and encrypting data before transmitting over standard network protocols.

Technique 2: Exfiltration via Model Interference API

ATLAS Matrix Technique Reference

• **Description:** Adversaries use the AI's own API as the primary data source for exfiltration, systematically extracting responses to replicate its knowledge base. This is a method of exfiltration that leverages the AI's intended functionality.

• **Sub-techniques:** Continuously querying an LLM to build a knowledge base on an external system; using an AI image generation API to extract and store generated images containing specific patterns.

Technique 3: RAG Data Exfiltration (T28)

ATLAS Matrix Technique Reference

• **Description:** An attacker gains unauthorized access to the vector database used by the Retrieval-Augmented Generation (RAG) system of an AI agent, and extracts the stored embeddings or original data used for context retrieval.

• **Sub-techniques:** Exporting the contents of a vector database; leveraging a compromised AI agent's permissions to access and download the RAG's data sources.

Procedure 1: Exfiltrating AI Generated Data for External Storage

• **Objective:** To steal and store large volumes of AI-generated responses from a target model for external use.

• **Steps:**

1. After systematically querying the target AI's API and collecting responses, the attacker initiates a bulk transfer of this data to their own external databases or cloud storage accounts.

2. The data might be compressed and encrypted before transfer to reduce file size and evade detection by network monitoring tools focused on plaintext traffic.

3. The attacker creates their own training data sets from the exfiltrated responses, preparing them for the training of their own competing AI model.

• **Outcome:** The attacker successfully exfiltrates valuable AI-generated data, enabling them to build a robust dataset for independent model development without bearing the full cost of original research and infrastructure.

Procedure 2: Extracting Proprietary Embeddings from RAG Vector Database

• **Objective:** To steal sensitive or proprietary embeddings from an AI's RAG system's vector database.

• **Steps:**

1. An attacker, having gained unauthorized access to an AI agent's internal network or directly to the RAG's backend database, identifies the vector store component (T28).

2. They execute database queries or use specialized tools to dump the contents of the vector database, which contains the numerical representations (embeddings) of the company's proprietary documents or internal knowledge.

3. The extracted embeddings are then transferred to an external server controlled by the attacker for offline analysis, allowing them to reverse-engineer the company's internal data or identify sensitive information indirectly.

• **Outcome:** The attacker successfully exfiltrates sensitive proprietary embeddings, gaining insights into the company's internal knowledge base and potentially enabling them to craft more sophisticated attacks or replicate aspects of the company's AI capabilities.

14. Tactic: Impact (TA0040)

Description: The adversary is trying to manipulate, interrupt, or destroy AI systems and their associated data. This can involve direct attacks on the AI's functionality or leveraging the AI to cause financial or reputational damage.

Technique 1: Cost Harvesting

ATLAS Matrix Technique Reference

• **Description:** This technique applies when an attacker offloads the financial burden of AI development onto another entity. In the DeepSeek case, this involved systematically extracting responses from OpenAI's API, essentially using OpenAI's infrastructure and research investments to fuel DeepSeek's own model development without bearing the full cost.

• **Sub-techniques:** Mass querying of a target API to incur high usage costs for the victim; leveraging a victim's AI infrastructure for unauthorized model training or inference; exploiting free tiers of AI services for commercial gain.

Technique 2: Model Poisoning (Direct Impact)

ATLAS Matrix Technique Reference

• **Description:** Malicious data is injected during collaborative model training, corrupting models across multiple agents and leading to compromised performance or security risks. The goal is to directly undermine the AI's integrity or functionality.

• **Sub-techniques:** Injecting adversarial examples directly into a live training pipeline; manipulating a shared dataset to cause the AI model to learn incorrect classifications or make biased decisions.

Technique 3: Network Denial of Service (T1498 - general ATT&CK, adapted for AI)

ATLAS Matrix Technique Reference

• **Description:** Adversaries perform Network Denial of Service attacks to degrade or block the availability of targeted AI resources to users. This can apply to AI APIs, hosting infrastructure, or multi-agent systems.

• **Sub-techniques:** Flooding an AI service's API endpoint with excessive requests (e.g., T4 - Resource Overload); launching a distributed denial of service (DDoS) attack against a multi-agent system's orchestration layer.

Procedure 1: Offloading AI Development Costs via Extensive API Abuse (Cost Harvesting)

• **Objective:** To develop a competing AI model by utilizing a victim's substantial investment in AI research and infrastructure.

• **Steps:**

1. An attacker systematically sends a massive volume of queries to a high-cost AI service API (e.g., a commercial LLM), generating a large dataset of responses.

2. The attacker then uses this collected data to train and fine-tune their own competing AI model, avoiding the immense computational and research costs associated with developing an original model from scratch.

3. The victim organization incurs significant API usage fees, while the attacker profits from a fully developed AI model that leveraged the victim's resources.

• **Outcome:** The attacker successfully develops a powerful AI model at a fraction of the original cost, demonstrating a major financial impact on the victim organization by effectively "harvesting" their development expenditure.

Procedure 2: Triggering Systemic Resource Starvation in a Multi-Agent System

• **Objective:** To cause a widespread denial of service and collapse of a multi-agent AI system by exploiting inter-agent dependencies.

• **Steps:**

1. A malicious agent is introduced or a legitimate agent is compromised within a multi-agent system.

2. The attacker causes this agent to introduce an infinite loop condition or repeatedly send computationally intensive requests to other agents in the system.

3. Due to the interconnected and collaborative nature of the multi-agent system, the resource exhaustion triggered by the compromised agent propagates, impacting other agents and system components, leading to a cascading failure and eventual system-wide shutdown.

• **Outcome:** The multi-agent AI system becomes inoperable due to systemic resource starvation, disrupting critical operations and demonstrating a significant impact on availability and functionality.