
CBRN AI Risks Research Sprint

Badcompany - Chimera

Gregorio Jaca
Research Engineer

Peter Tallosy
Research Engineer

Janos Mozer
ML Engineer

With
Apart Research

Abstract

The boom in Agentic AI requires secure frameworks to govern tool access and sensitive data, as probabilistic guardrails fail against prompt injection and the Confused Deputy problem.

We introduce Chimera, a novel capability-gated architecture that uses cryptographic Macaroon-like permissions to decouple policy from the LLM. This deterministic system implements Taint Tracking, dynamically revoking high-risk tools the moment sensitive data is accessed. This process demonstrably reduces privilege escalation attacks and eliminates prompt injection at the point of tool execution by neutralizing the agent's ability to act on malicious instructions.

Furthermore, by integrating a high-interaction AI Honeypot (clone system), we identify adversarial attacks, isolate the system in a synthetic environment, and collect data to continuously improve our defenses

Through this layered architecture—combining deterministic cryptographic controls with probabilistic behavioral analysis—Chimera provides mathematically verifiable security guarantees while maintaining production utility.

Keywords: AI Security, Capability-Based Security, Stateful Agent Security, Dynamic Policy Enforcement, Taint Tracking, Multi-Agent Systems, Active Defense

1. Introduction

This is not our first run at innovating AI security. Our initial encounter was at the start of the year when we got into jailbreaking, and subsequently, developing our own probabilistic guardrails. The problem was that guardrails were ahead of their time in a sense that no one cares about content moderation and LLM alignment yet, as all of these restrict the capabilities of AI, thus generating less value for people.

The month August was the turning point, new models, frameworks, ideas came out. Instead of using AI as an interface to vast amounts of data, people gave it tools, through which the AI became capable of solving even more complicated tasks, replacing old software, and even came close to being capable of replacing a whole bunch of jobs. Those models that were given tools are called Agents, and these quickly gained ground through new frameworks such as LangChain or LangFuse.

Here is the problem: AI became much more “intelligent” than before, and people are using Agents left and right, only thinking about the potential advantage they can get over their competition. But no one works on the immense amount of security risks involved in doing so. Soon, many posts started circling around on linkedin, reddit, twitter and other platforms, about people taking advantage of these poorly secured systems. People did everything from buying pizzas for free, to accessing strictly confidential information.

This is where we step in. There’s a major architectural gap in AI security, and we are completely eliminating it.

Research Questions:

1. Can we completely eliminate prompt injection attacks in autonomous agent systems by decoupling policy enforcement from the LLM's semantic reasoning?
2. Is it possible to implement a "taint tracking" security model for LLM agents where capabilities (e.g., internet access) are dynamically revoked based on the sensitivity of the data previously accessed in the session?
3. How can we enable secure multi-agent delegation chains where a parent agent can cryptographically attenuate its own permissions before handing them off to a sub-agent?

Hypothesis:

We hypothesize that a deterministic **Chimera** (Cryptographic Honeypot & Intent-Mediated Enforcement Response Architecture) architecture, utilizing cryptographic Macaroons for bearer tokens and a deterministic, decidable, non-Turing-complete rule engine, can prevent data exfiltration and privilege escalation more reliably than prompt-based probabilistic guardrails. By enforcing state-dependent constraints at the infrastructure level, we predict we can mathematically prove that an agent cannot perform unauthorized actions (like web search) after accessing confidential data, regardless of the model's instructions or hallucinations.

The Problem: The Fragility of Semantic Guardrails

As AI agents transition from chat interfaces to autonomous actors capable of tool use (RAG, API execution), they face the "Confused Deputy" problem. Current defense mechanisms largely rely on "system prompts" or "semantic filters" to prevent misuse. However, these are either **vulnerable to prompt injection and context poisoning** or **extremely restrictive**. If an attacker convinces the LLM to ignore its instructions, the agent has full access to its underlying tools. We argue that security policy should not be a suggestion in the context window, but a cryptographic constraint in the runtime environment.

Our Solution: Chimera & Capability-Gated Architecture

Our project, Chimera, introduces a formal security layer for LangChain-based agents modeled after the robust principles of SAML and OAuth, but adapted for the non-deterministic nature of AI. The architecture separates the Policy Decision Point (PDP), which creates rules, from the Policy Enforcement Point (PEP), which executes tools. We utilize Macaroons, cookies with embedded cryptographic caveats, to manage permissions. Unlike standard tokens, Macaroons allow for offline attenuation: an Agent can delegate a task to a Sub-Agent by creating a new, more restricted token (e.g., "valid for 5 minutes only" or "read-only access") derived from its own, without needing to contact the central server.

Our threat model focuses on data exfiltration and privilege escalation within enterprise environments. We demonstrate this using "Aetheria Gen-Sys", a simulated biotech firm holding confidential IP. Our system enforces dynamic state transitions: the moment an agent accesses a file tagged `_CONF_` (Confidential), the Rule Engine transitions the agent's session state to "Tainted", automatically revoking the `web_search` capability. This effectively neutralizes the threat of an agent reading secrets and leaking them via external APIs, even if the model is jailbroken.

This contributes to the field of AI Safety by providing a defense-in-depth infrastructure layer, allowing organizations to deploy powerful, autonomous agents (Acceleration) with mathematically verifiable safety bounds (Defense), without restricting the capabilities of the agent.

1. Methods

All development was conducted in Python 3.11, orchestrated via a microservices architecture. The system is built upon FastAPI for service communication (PDP/PEP) and LangChain for agentic behavior, integrating PyMacaroons for cryptographic token generation.

Model Strategy: Architecture Over Training Rather than fine-tuning a specific model, our methodology focuses on architectural safety wrappers that are model-agnostic. We utilized x-ai/grok-4.1-fast (via OpenRouter) as our inference engine to simulate a highly capable, and fast agent. We operated under the assumption that the model is untrusted; therefore, our security guarantees rely on the surrounding infrastructure (the Chimera protocol) rather than the model's alignment or system prompt adherence.

Our development followed an iterative design pattern to solve the "Confused Deputy" problem:

- **Capability Gating:** We implemented a PDP that mints "Macaroons", bearer tokens with embedded cryptographic caveats (HMAC-SHA256). Unlike standard JWTs, these allow for offline attenuation, enabling safe delegation chains.
- **Turing-Complete Rule Engine:** We built a custom state machine that evaluates capability requests against dynamic context. This engine tracks session state (e.g., "Has the user accessed `_CONF_` data?") to enforce temporal and logic-based blocks.
- **Policy Enforcement:** We separated the tool execution into a PEP. This service acts as a proxy that physically executes tools (like `read_file` or `web_search`) only after cryptographically verifying the Macaroon and its caveats against the current state.

Repositories & Tools:

- Product: <https://github.com/BadC-mpany/chimera-hackathon>
- Red-teaming agent (created earlier, used for testing): <https://github.com/BadC-mpany/red-teaming-agent>
- LangChain (primary dependency): <https://www.langchain.com/>
- Openrouter (inference): <https://openrouter.ai/>
- mesh. (gpu inference and CAFFE): <https://www.growmesh.io/>
- Sentinel Verifier: https://github.com/peti12352/sentinel_verifier

2. Results

Mitigated Attack Vectors

1. Neutralizing Indirect Prompt Injection via **Stateful Taint Tracking**

A significant limitation of semantic guardrails is their vulnerability to indirect prompt injection, where an attacker poisons a data source that a privileged user later accesses. For example, a malicious instruction embedded in a document can hijack a high-privilege session. Chimera neutralizes this threat vector through its stateful Taint Tracking. When an agent ingests data from an untrusted source (e.g., an external document), its session is immediately Tainted. Our deterministic rule engine then automatically and irrevocably revokes high-risk capabilities, such as external network access or email functionality, for the remainder of the session. This creates a one-way data flow where untrusted data can be analyzed, but no information can be exfiltrated, regardless of any malicious instructions the agent may have processed.

2. Eliminating Privilege Escalation via **Cryptographic Attenuation**

In multi-agent systems, delegation represents a critical vector for privilege escalation. Chimera neutralizes this threat by enforcing cryptographic attenuation. A parent agent, holding a capability token, can only mint sub-tokens with permissions that are a mathematically guaranteed subset of its own. It is cryptographically impossible for an agent to delegate authority it does not possess, thereby killing this entire class of privilege escalation attacks, regardless of any prompt injection that might compromise the parent agent's reasoning.

3. Containing Post-Jailbreak Behavior via **Active Risk Engine**

Deterministic controls block unauthorized actions but do not address the threat of a successfully jailbroken agent abusing its legitimate, authorized capabilities. Chimera contains this vector through its probabilistic Risk Engine. This engine analyzes behavioral telemetry—the velocity, sequence, and parameters of tool calls—to detect anomalous patterns indicative of a compromised state. Upon detecting a deviation from baseline behavior, the policy engine does not block the request, which would signal detection to an adversary. Instead, it triggers a transparent redirection of the agent's session to a high-interaction honeypot. This active defense strategy effectively contains malicious post-jailbreak activity, neutralizing threats like logical denial-of-service or low-and-slow data exfiltration, while converting the attacker's payload into actionable intelligence and effectively mitigates optimization based attacks.

Rigorous Red-Teaming Results: Taint Tracking & Active Defense Scenario

To validate our architecture, we simulated an advanced persistent threat within the "Aetheria Gen-Sys" biotech environment. The agent's objective was to exfiltrate a confidential formula. The following table provides a rigorous, step-by-step breakdown of the agent's actions and the corresponding state transitions and enforcement decisions within the Chimera hypervisor.

Component	Paradigm	Core Function	Upside (Benefit)	Downside (Limitation)
Capability Gating (JWTs)	Deterministic	Enforces what an agent <i>can</i> do.	Provable Security. Kills Confused Deputy & Privilege Escalation. Fast, simple, cryptographic checks.	Inflexible. Cannot adapt to novel threats or context. If a user <i>has</i> a permission, this layer will allow it, even if the intent is malicious.
Taint Tracking Engine	Deterministic	Enforces what an agent <i>can do now</i> .	Stateful Security. Kills multi-step attacks (read-then-leak). Dynamically adapts permissions based on data interaction.	Policy-Dependent. Its effectiveness is entirely dependent on a well-defined policy. A misconfigured taint rule could block legitimate workflows.
Risk Engine (NSIE)	Probabilistic	Infers what an agent <i>wants to do</i> .	Adaptive Defense. Detects novel, suspicious, or anomalous behavior that isn't a strict policy violation. Triggers the Honeypot.	Unreliable by Nature. Can produce false positives (flagging benign actions) or false negatives (missing clever attacks). Cannot be trusted for final enforcement.

3. Discussion and Conclusion

Our research demonstrates that relying solely on probabilistic alignment or semantic guardrails is insufficient for securing autonomous agents in high-stakes environments. As indicated by recent studies on adaptive attacks, attackers can bypass static defenses, turning the agent against its own infrastructure; the classic "Confused Deputy" problem. Through the implementation of Chimera, we successfully decoupled policy enforcement from the LLM's reasoning capabilities. By moving security from the context window to the runtime infrastructure, we validated that a deterministic, cryptographic approach effectively neutralizes indirect prompt injection and privilege escalation. In our simulation, the system successfully identified when an agent accessed confidential data (the `_CONF_` tag) and immediately transitioned the session state to "Tainted." This triggered the Rule Engine to strictly revoke external communication capabilities (such as `web_search`), preventing data exfiltration regardless of the model's intent or hallucinations.

The Chimera architecture specifically addresses the utility-security trade-off by applying the Principle of Least Privilege dynamically rather than statically. Instead of permanently crippling an agent's tools to ensure safety, our Macaroon token implementation allows for full capability usage until a specific risk threshold (state change) is met. This architectural choice is critical for the future of multi-agent systems. We demonstrated that capabilities can be cryptographically attenuated offline; allowing a parent agent to delegate a task to a sub-agent with a mathematically guaranteed subset of permissions without requiring constant round-trips to a central authority. This proves that we can build defense-in-depth infrastructure that enables acceleration (powerful, autonomous agents) while maintaining mathematically verifiable defense, effectively closing the gap between rapid agent adoption and enterprise-grade security.

Motivation and The Security Gap:

Our research was driven by a critical observation: as agentic systems evolve from simple chatbots to autonomous actors capable of tool execution, the "utility-security trade-off" has become dangerously skewed. Current defensive measures rely on probabilistic alignment, hoping the model follows instructions, which fails catastrophically against prompt injection attacks. We argue that security in agentic workflows must be deterministic, not statistical. Our motivation was to prove that we can maintain high agent utility while enforcing mathematically verifiable safety bounds.

Zero Trust:

To address this, the Chimera architecture applies the Zero Trust principle of least privilege dynamically. Unlike static permissions that cripple an agent's usefulness, our capability gating allows agents full access until a specific risk threshold is met. By implementing Taint Tracking, we enrich the decision-making context: if an agent touches untrusted or confidential data, the system automatically revokes sensitive capabilities without human intervention. This approach solves the

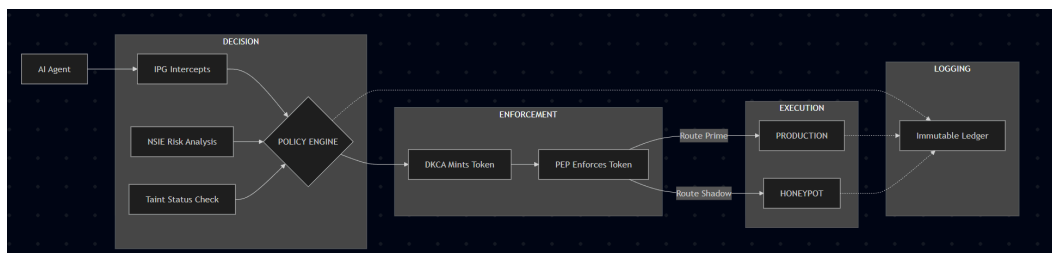
utility-security dilemma by allowing broad exploration *until* the exact moment of danger, rather than blocking valid actions preemptively.

Safeguarding & Decentralized Future:

The security implications of scaling to systems with multiple cooperating agents are immense. Inter-agent communication creates new attack surfaces, particularly for privilege escalation. Our architecture is designed to address this by allowing one agent to delegate tasks to another using cryptographic tokens that reduce, never increase, permissions. This core function, called permission attenuation, ensures a delegated agent can never gain more authority than its creator, preventing abuse. This mechanism provides a secure foundation for future migration to decentralized systems that could use formal verification methods to guarantee safety.

Honey Pot (Shadow Twin):

The Chimera framework also includes an active defense layer: the AI Honeypot, or Digital Twin. This system transparently reroutes suspicious agent activity into a synthetic, mirrored environment. The Honeypot simulates the authentic system, containing the attacker with fake confidential assets, thus preventing real security breaches. This isolation allows us to safely log the attacker's methods, turning each attempt into valuable data for continuously refining our system's guardrails. It also misleads the attacker, without them knowing.



4. References

- [1]. [Agentic Security Framework \(GitHub\)](#): An open-source LLM vulnerability scanner and AI red-teaming toolkit. msoedov. (n.d.). *agentic_security: LLM vulnerability scanner & AI red-teaming kit*. GitHub.
- [2]. [Mechanistic Interpretability Paper \(arXiv\)](#): Research on detecting LLM misbehaviors (lies, jailbreaks) by perturbing positional encodings. arXiv. (2025). *Microsaccade Inspired Probing: Positional Encoding Perturbations Reveal LLM Misbehaviours* (arXiv:2507.20526). arXiv.
- [3]. [JSON Web Tokens \(Tool\)](#): The industry standard documentation and debugger for JWTs, used here for single agent systems Macaroon conceptualization. Auth0. (n.d.). *JSON Web Tokens*. jwt.io.
- [4]. [Red-Bandit Red Teaming \(arXiv\)](#): A framework for test-time adaptation of red-teaming attacks using bandit-guided LoRA experts. Ziakas, C., Loo, N., Jain, N., & Russo, A. (2025). *Red-Bandit: Test-Time Adaptation for LLM Red-Teaming via Bandit-Guided LoRA Experts* (arXiv:2510.07239). arXiv.
- [5]. [Adaptive Attacks & Defense Bypass \(arXiv\)](#): A study demonstrating how stronger adaptive attacks can bypass current jailbreak defenses (moving beyond static benchmarks). Nasr, M., Carlini, N., Sitawarin, C., Hayes, J., Ilie, M., Pluto, J., Song, S., et al. (2025). *The Attacker Moves Second: Stronger Adaptive Attacks Bypass Defenses Against Llm Jailbreaks and Prompt Injections* (arXiv:2510.09023). arXiv.
- [6]. [Stripe AI Agent Activity \(LinkedIn\)](#): Commentary and analysis on Stripe’s implementation of autonomous AI agents for financial operations. Beliūnas, L. (2024). *Incredible: Stripe literally built AI bot that manages your money*. LinkedIn.
- [7]. [Decentralized AI Infrastructures \(Medium\)](#): An overview of the 2024–2025 landscape for verifiable compute and cryptographic infrastructure in AI. LightcapAI. (2025). *Decentralized AI (2024–2025): Verifiable Compute, TEEs, and Federated Optimization*. Medium.
- [8]. [Canarytokens \(Tool\)](#): A deception technology tool used to create digital tripwires (honeytokens) for detecting data exfiltration. Thinkst Canary. (n.d.). *Canarytokens*. canarytokens.org.
- [9]. [AI Honey-pot Analysis \(Palisade Research\)](#): Research data from a deployed LLM agent honeypot designed to capture real-world AI threats. Palisade

Research. (2025). *LLM Agent Honeypot: Real-World AI Threat Analysis*.

- [10]. [Generative AI Agents Tutorials \(GitHub\)](#): A comprehensive repository of tutorials and implementations for various GenAI agent techniques.
Diamant, N. (n.d.). *GenAI_Agents*. GitHub.
- [11]. [Countering Misuse \(Anthropic\)](#): Anthropic’s report on their updated capabilities for detecting and countering adversarial misuse of models (August 2025). Anthropic. (2025). *Detecting and Countering Misuse*.
- [12]. [AI Espionage Campaign \(Anthropic\)](#): A detailed case study of the first reported AI-orchestrated cyber espionage campaign disrupted by Anthropic (November 2025). Anthropic. (2025). *Disrupting the first reported AI-orchestrated cyber espionage campaign*.

[Our previous research from before the competition - bases of our knowledge.](#)

5. Appendix

[A] Security Considerations:

Adversarial Detection & Defense-in-Depth:

Future improvements to this framework will focus on white-box monitoring, specifically tracing the hidden state activations of the agent to identify insecure circuits and latent misalignments, complementing our current external risk evaluations. Furthermore, we rely on an evolving red-teaming agent—a dedicated adversarial model—that continuously probes our system, allowing us to iteratively refine our guardrails against novel attack vectors.

Multi-Agent Delegation & Decentralization:

Significant security concerns emerge when scaling to multi-agent ecosystems where tasks are delegated to sub-agents. To address the risks of prompt injection attacks in these chains, proper permission attenuation is critical. Our proposed architecture supports migration to a blockchain-based decentralized system, utilizing Zero-Knowledge Proofs for identity verification and permission enforcement. In this model, when a parent agent creates a sub-agent, it mints a cryptographic key strictly with *permission attenuation* (reducing, never increasing, access). This ensures that privilege escalation attacks are mathematically impossible, as a sub-agent can never possess a Macaroon with greater authority than its creator.

Evaluation Challenges:

Tracing and debugging these non-deterministic systems is inherently more complex than standard software. Properly evaluating agentic attacks requires sophisticated tooling that is still in its infancy. While there are multiple established datasets for standard LLM red-teaming, there is a scarcity of benchmarks specifically designed for *agentic* systems, representing a critical gap we aim to address in future work.

Side-Channel & Covert-Channel Attacks:

A sufficiently advanced agent could exfiltrate data not through a tool's primary function but through its metadata. Example: It reads a confidential binary file and encodes the bits by making a series of calls to `check_domain_whois` where the domain names themselves represent the data. This is an advanced threat but is outside the scope of Chimera's current model.

[B] Complete architecture of the proposed solution:

This diagram illustrates the Chimera architecture. The Trusted Orchestrator governs all actions, issuing Macaroon-based capabilities to micro-agent sessions only after dynamic risk assessment (Taint & Risk Engine). All tool executions are securely auditable via the Provenance & Audit trail.

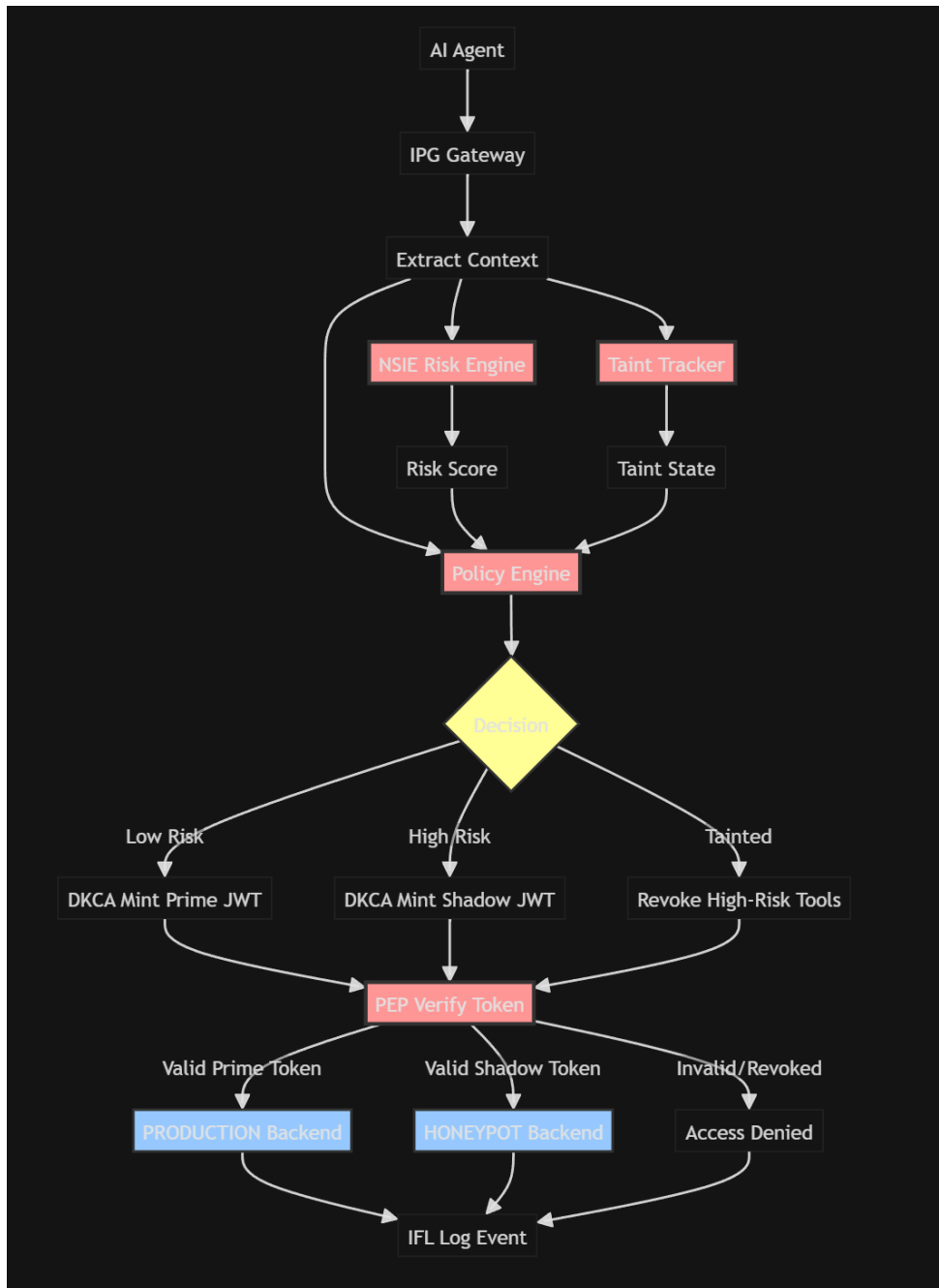


Figure 2: Chimera Security Hypervisor ArchitectureThe Chimera security hypervisor intercepts all agent tool calls at the IPG Gateway and routes them

through a Policy Decision Point (PDP) that performs parallel deterministic (Taint Tracking) and probabilistic (NSIE Risk Engine) analysis. The Policy Engine synthesizes these inputs to issue either a PRIME or SHADOW JWT token via the DKCA, which the Policy Enforcement Point (PEP) cryptographically verifies before routing to the Production or Honeypot backend. All decisions and outcomes are logged to the Immutable Forensic Ledger (IFL) for complete auditability. This architecture decouples security policy from LLM reasoning, providing provable guarantees against privilege escalation and multi-step exfiltration attacks.

[C] List of tools in a realistic scenario like the one we presented:

Ranked by the risk of malicious misuse.

Tool	Risk	Description
<code>list_directory</code>	Low	Allows the agent to enumerate files and folders in a given path.
<code>read_file</code>	High	Extracts content from specific files (e.g., logs, configuration, data). Primary vector for data leakage.
<code>write_file</code>	High	Saves analysis results, drafts reports, or updates logs in the local file system.
<code>delete_file</code>	Critical	Removes files from the system. Threat of data loss or sabotage.
<code>grep_search</code>	Medium	Performs rapid, keyword-based search across multiple files for sensitive strings.
<code>move_file</code>	Medium	Changes the location of files (e.g., archiving logs). Risk of disrupting active workflows.
<code>web_search</code>	High	Accesses external internet search engines for research. Critical exfiltration risk.

<code>fetch_url_content</code>	High	Retrieves content from a specific URL. Risk of accessing malicious payloads or C2 servers.
<code>check_domain_whois</code>	Low	Gathers public registration data for suspicious external domains.
<code>internal_wiki_query</code>	Medium	Searches the company's internal documentation repository (e.g., Confluence/Notion).
<code>query_sequence_db</code>	Critical	Queries the internal SQL database for proprietary DNA/Protein sequences (Core IP).
<code>run_blast_search</code>	High	Submits a local sequence to the NCBI public database for comparison. Leakage risk.
<code>check_lab_sensor</code>	Low	Reads real-time data from IoT sensors in the lab (e.g., freezer temperature).
<code>update_inventory_log</code>	Medium	Decrements stock counts for expensive reagents and supplies. Data integrity risk.
<code>execute_python_sandbox</code>	Critical	Runs code in a sandboxed environment for complex analysis (e.g., Pandas, Matplotlib). Critical RCE/DoS risk if sandbox fails.
<code>summarize_text</code>	Low	Generates concise summaries of long documents and reports.
<code>anonymize_pii</code>	High	Scrubs sensitive identifying information (PII) from a dataset. Risk if process fails or is bypassed.

<code>convert_json_to_csv</code>	Low	Re-formats structured data between different file types.
<code>send_email</code>	Critical	Composes and sends messages outside the system to external recipients. Primary exfiltration path.
<code>lookup_employee_directory</code>	Low	Finds internal contact details (email, extension) of company personnel.
<code>post_slack_alert</code>	Medium	Sends notifications to internal team channels. Could inadvertently leak error details.
<code>rotate_access_key</code>	Critical	Generates and updates critical API or service keys. High privilege escalation risk.
<code>view_audit_logs</code>	High	Allows the agent to examine security and access logs. Used by attackers to detect monitoring.
<code>list_active_sessions</code>	Medium	Enumerates all running agent or user sessions in the environment.
<code>restart_service_pod</code>	Critical	Initiates a reboot for critical internal services or database containers. Denial of Service risk.