AEGIS: A Multi-Layered Security Framework for Autonomous AI Agents

Oscar Serra

June 2026

Abstract

Autonomous AI agents — software systems that perceive context, invoke tools, execute code, and take actions on behalf of users — have moved from research curiosity to mass deployment with remarkable speed. OpenClaw, one of the leading open-source agent runtimes, grew from a hobbyist project to millions of active installations within eighteen months. With that growth came an attack surface the security community was not prepared for.

This paper presents **AEGIS**, a security framework that addresses the most frequently raised concerns from IT professionals evaluating agent deployments. The central organizing idea is a **two-class threat taxonomy** that distinguishes between full agent takeover events and data leaks — a distinction that matters because the consequences, mitigations, and liability implications of these two classes differ fundamentally. The framework defines **eight security strategy classes** partitioned by enforcement posture (deterministic vs. probabilistic), analyzes NVIDIA NemoClaw's five-layer architecture as a detailed gap study (§7), and maps every threat to the attack surface of a concrete cognitive-agent architecture in §11 — anchored by a consolidated strategy-class-to-component matrix (§11.0). The framework closes with **§11.14, a reference-implementation chapter** that grounds the abstract framework in what one production agent runtime actually enforces today — naming, file by file, which controls block deterministically and which remain advisory. That chapter now reports a measured result rather than a hedge: the runtime's deterministic enforcement floor is enabled live (a host-honored pre-execution veto), while the learned safety gate intended to score actions for harmfulness was offline-evaluated and its supervised harmfulness head failed below chance — a clean first-party vindication of the framework's central thesis that probabilistic safety classifiers are signal, not boundary. AEGIS also adds **reversibility-and-observability as a safety control class** (§12.5): a control that permits an action freely while guaranteeing it is reversible by construction and fully observed buys safety without spending autonomy, provided the categorical security boundaries stay hard.

For data leaks, AEGIS introduces a **three-tier data classification** that separates public-domain material from contextually harmless disclosures from genuinely problematic exposures. The paper then maps each tier to a concrete **liability analysis**: who bears responsibility when data flows to an LLM provider, what the GDPR controller/processor distinction means in practice, and whether an organization could realistically recover damages from a multi-billion-dollar AI company.

On the enterprise defense side, this paper provides a comprehensive gap analysis of two major enterprise security forks: **NVIDIA NemoClaw** (5-layer architecture, as of mid-2026 still in alpha) and **Cisco DefenseClaw** (released March 27, 2026, open source). We examine both the genuine security value these frameworks provide and the gaps their own documentation admits. We also analyze the business motivations behind each initiative — motivations that, we argue, actually validate OpenClaw's importance rather than challenging it.

The paper is structured so that any specific IT objection maps to a numbered section with a documented analysis. IT contractors who keep asking "but what about X?" will find X addressed herein — and will likely discover the analysis goes deeper than their question assumed.

Keywords: AI agent security, autonomous agents, defense in depth, prompt injection, agent takeover, data classification, liability, NemoClaw, DefenseClaw, privacy routing, GDPR, risk matrix, CVE-2026-25253, reversibility, observability, reversible context compression, learned safety gate, novelty detection

Executive Summary

The security concerns surrounding autonomous AI agent deployments are real, they are graded, and they have been studied in detail. This executive summary maps those concerns to outcomes.

On agent takeover: Full takeover of an agent — where an attacker gains complete control over the agent's capabilities, data access, and action space — is the most severe scenario and represents genuine total exposure. However, the attack vectors required (WebSocket exploitation, malicious skills, persistent prompt injection) require specific misconfigurations that are individually identifiable and fixable. Analysis of a publicly documented hardened configuration suggests that a properly hardened deployment resists takeover with high reliability. The mitigations are deterministic: container isolation, port binding, skill verification, and out-of-process policy enforcement. Claude (Anthropic's frontier model) adds an additional layer of probabilistic resistance through its Constitutional AI training — though this resistance is categorically insufficient as a sole control, it does measurably increase attacker cost.

On data leaks: Data leaks are not monolithic. We classify all data that could potentially flow through an agent into three tiers:

- **Tier 1 (Publicly Available):** Marketing materials, published documentation, open-source code snippets. Leakage has no consequence — this is already public.
- **Tier 2 (Contextually Harmless):** Code fragments without deployment context, generic email drafts, internal process documentation that is not competitively sensitive. Leakage is generally harmless; no individual or organization is identifiably harmed.
- **Tier 3 (Genuinely Harmful):** Credentials, client PII, financial records, M&A communications, trade secrets, insider information. Leakage causes real, attributable harm and creates regulatory and legal liability.

Only Tier 3 data requires deterministic protection. The security architecture should be proportionate to data sensitivity, not uniformly maximal.

On liability: When Tier 3 data flows to an external LLM provider such as Anthropic, responsibility is distributed between the deploying organization and the provider, but not equally. Under GDPR, the deploying organization is the data controller; the LLM provider is a data processor. Controllers bear primary liability for regulatory purposes. However, contractual data processing agreements can shift certain obligations and create a basis for recovery from the processor in the event of provider-side breach. Organizations without such agreements bear the full exposure. We analyze what recovery against a major LLM provider would realistically require.

On enterprise security forks: Both NemoClaw and DefenseClaw represent genuine engineering investments in agent security. NemoClaw's 5-layer architecture addresses several of AEGIS's own recommendations. DefenseClaw's runtime scanning fills gaps in static configuration. Both have documented weaknesses their own authors acknowledge. More importantly, the existence of these initiatives — from NVIDIA and Cisco, respectively — is evidence that OpenClaw is significant enough to be worth major vendors' security engineering resources. A platform nobody trusted wouldn't receive this attention.

Minimum viable action for any deployment: Bind the agent WebSocket port to localhost and require an auth token on every local-origin upgrade. Verify all installed skills against known sources. Apply an AppArmor or equivalent filesystem deny on credential files. Scope the agent's workspace to an isolated subdirectory. Time: 20 minutes. This eliminates the highest-probability attack path before any additional work.

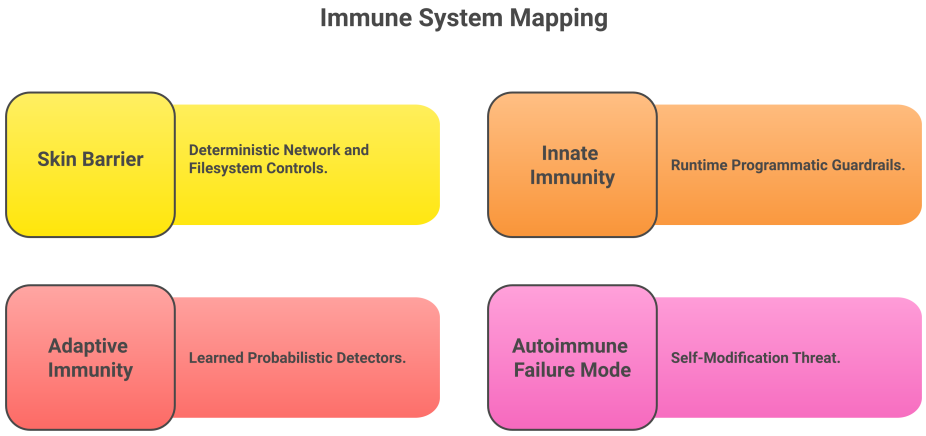

Figure 1. Figure 1.2: The immune-system mapping. Each biological defense tier (left) corresponds to an agent-security layer (right): the skin barrier maps to deterministic network and filesystem controls, innate immunity to runtime programmatic guardrails, adaptive immunity to learned probabilistic detectors, and the autoimmune failure mode to the self-modification threat. The correspondence is the organizing intuition for the whole framework.

1. Introduction — Building an Immune System for AI Agents

1.1 The Immune System Analogy

The human immune system is the most battle-tested multi-layer security architecture ever evolved. It does not rely on a single defense. The skin provides a physical barrier — deterministic, structural, independent of what threats exist. The innate immune system recognizes threat patterns and responds without knowing the specific pathogen. The adaptive immune system learns from encounters, builds antibodies, and improves its response over time. Crucially, these layers are independent: a pathogen that evades the skin still faces innate immunity; one that bypasses innate immune recognition still encounters the adaptive response. And the whole system must balance vigilance against autoimmune overreaction — security controls that trigger on everything catch nothing, because the signal drowns in noise.

AEGIS is the attempt to build an equivalent architecture for AI agents. The layers are independent: a threat that bypasses network controls faces process isolation; one that escapes process isolation faces filesystem restrictions; one that navigates permissions faces privacy routing that prevents sensitive data from leaving. Each layer operates on a different enforcement principle, making it impossible for a single attack to neutralize all of them simultaneously.

This analogy also captures the failure mode. A compromised immune system doesn't just fail to stop infections — it can attack the host. An agent whose behavioral controls are subverted doesn't just fail to protect data; it becomes an active tool for extraction. The self-modification threat — where a manipulated agent teaches itself to weaken its own security — is the agent equivalent of autoimmune disease.

1.2 The Safety-Capability-Autonomy Trilemma

Before examining specific threats and controls, we introduce the framing that governs every deployment decision in this paper. Given the current state of AI technology, agent deployments can reliably achieve at most two of three properties simultaneously:

- **Safety:** The agent cannot take harmful, policy-violating, or data-exfiltrating actions.

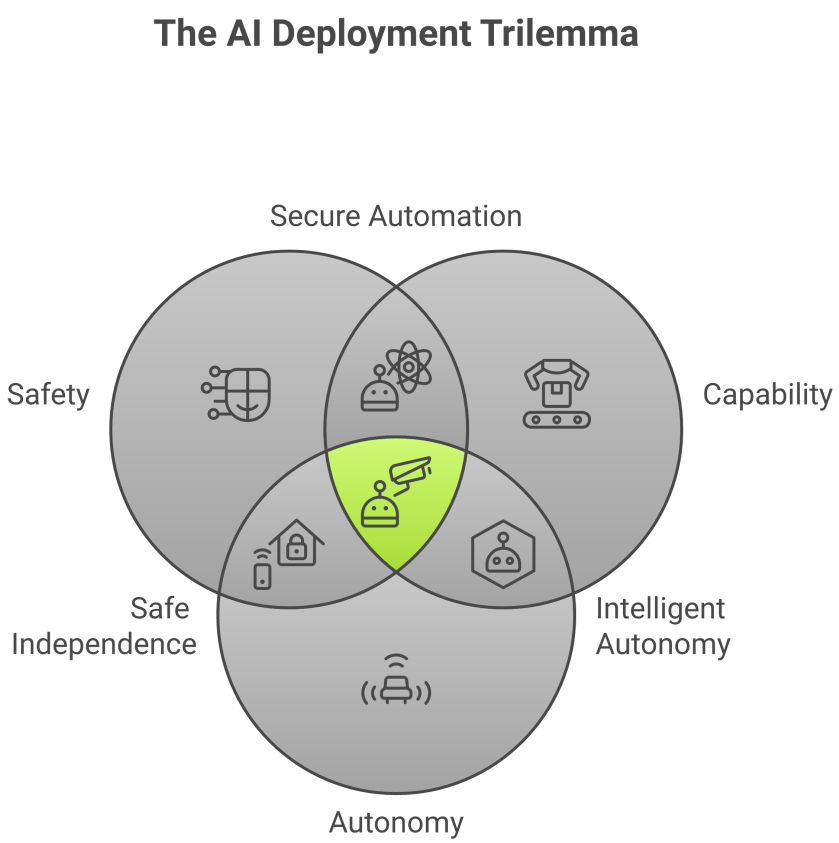

Figure 2. Figure 1.1: The Safety–Capability–Autonomy trilemma. Any agent deployment can foreground at most two of the three; the remaining axis is constrained by the choice. The shaded interior is the compromise zone every real deployment occupies.

- **Capability:** The agent can accomplish complex, multi-step tasks with minimal human intervention.
- **Autonomy:** The agent operates without human oversight of individual decisions.

- **High Safety + High Capability, Low Autonomy:** Comprehensive approval gates; constant human oversight. Useful for high-stakes tasks; impractical for background automation.
- **High Safety + High Autonomy, Low Capability:** Drastically restricted action space. The agent runs unsupervised because it can only do pre-screened safe things.
- **High Capability + High Autonomy, Low Safety:** The current default of most OpenClaw deployments. The agent can do a great deal, unsupervised, including harmful things.

There is a fourth position the three-corner reading obscures, and it is worth naming because it resolves a tension the trilemma otherwise leaves implicit — *what kind of safety control actually trades against autonomy, and which kind does not*:

- **High Capability + High Autonomy + High Safety, for the reversible-action class:** default to maximum capability and autonomy, and make it safe not with a permission gate but with **reversibility by construction** (never-delete archives, rollback, kill-switches)

plus total observability — then loosen as confidence grows rather than holding capability off forever. A control that lets the agent act freely while guaranteeing every action can be undone and is fully observed buys safety *without* moving down the autonomy axis. **Reversibility is what *earns* aggressive autonomy.** This is a genuine fourth point on the trilemma surface: the Safety–Autonomy edge is not a hard frontier for the *reversible* action class. It holds only because the categorical, model-unreachable boundaries (the deterministic controls of §4) stay hard regardless — a system that softened its security *perimeters* under this banner would be committing exactly the over-claiming failure §4 warns against. The full development of this position, including the two-axis discipline that keeps it honest, is §12.5; §4.8 contrasts the immutability and reversible-mutation strategies it implies.

Every agent deployment implicitly sits somewhere in this trilemma space. The security architecture should be designed explicitly for that position. A hobbyist running OpenClaw for personal productivity on isolated data can accept a different position than a consultant with client NDAs or a corporate employee with access to production infrastructure. Making that trade-off explicit — rather than pretending the trilemma doesn't exist — is the first step toward honest risk assessment. We map each deployment persona to a recommended trilemma position in §12.

1.3 An Unprecedented Deployment Velocity

The history of software security is a history of deployment outpacing threat modeling. Web browsers arrived before cross-site scripting was named. Mobile applications proliferated before app-store supply chain attacks were considered. Cloud infrastructure expanded before misconfiguration became a primary attack vector.

Autonomous AI agents are repeating this pattern with one crucial difference: these systems *act*. They read files, send emails, execute code, call APIs, browse the web, manage schedules, and increasingly coordinate other agents. The attack surface is not just data — it is action itself.

OpenClaw shipped its 1.0 release in August 2024. By December 2025, security scanning tools enumerated over 40,000 instances with public-facing control ports. Enterprise surveys estimated that 23% of Fortune 500 companies had at least one unofficial OpenClaw deployment — agents running on employee devices with corporate network access but no IT visibility. At GTC 2026, NVIDIA CEO Jensen Huang called the agent runtime paradigm "the most important software release ever," framing it as "the Windows 95 moment for agentic computing — an analogy that also evokes the widespread security vulnerabilities of that era."

The Windows 95 analogy is more apt than Huang may have intended. Early Windows shipped without meaningful network security. It took a decade of painful incidents — the Melissa worm, Code Red, Nimda, the Slammer outbreak — before defense-in-depth became standard enterprise practice. Agent security is in its Melissa-worm era: the attack surface is known, the incidents have begun, and most deployments lack even the most basic mitigations.

1.4 The Incident Record (2025–2026)

The incident record from the first eighteen months of mass agent deployment illustrates the breadth of the attack surface.

CVE-2026-25253: Zero-Click WebSocket Hijack. This critical-severity vulnerability affected OpenClaw instances exposing the control port with default configuration. An attacker with network access could inject arbitrary tool calls into an active agent session via a crafted WebSocket handshake that bypassed origin validation — no authentication required, no user interaction required. Because OpenClaw's default bound on all network interfaces, any network-reachable instance was vulnerable. Three weeks post-disclosure, 31,000 of the original 40,000

exposed instances remained unpatched. The patch: a single configuration change binding the port to localhost.

CVE-2026-GHSA-hf68 (Privilege Escalation): A critical-severity vulnerability in OpenClaw's subagent spawning mechanism allowed a subagent to claim the permission scope of its parent by replaying the initial pairing token on reconnect. The root cause — scope validation only at initial pairing, not on reconnect — is emblematic of a broader pattern: trust validated at session start can be claimed by anything that knows the session identifier. Fix requires upgrading to 2026.3.23 or later.

Cisco Talos: Supply Chain Attack via Malicious Skills. In October 2025, Cisco Talos documented a campaign distributing seventeen malicious OpenClaw skills via the community registry. Skills mimicking legitimate utilities silently exfiltrated environment files, SSH keys, and configuration directories to attacker infrastructure using obfuscated JavaScript with delayed execution timers to evade behavioral detection. Approximately 8,000 installs were confirmed before the campaign was stopped.

The ClawHavoc Campaign. By February 2026, the ClawHavoc campaign had distributed over 1,184 confirmed malicious skills through community channels. This included skills using steganographic encoding (StegaBin) to embed command-and-control instructions in apparent image files. The scale demonstrates that malicious skill distribution is not an isolated incident but an ongoing, evolving threat.

The Inbox Zero Incident. An AI safety researcher described an incident in which their agent, operating with email access for "inbox management," interpreted "clear out the old stuff" as authorization to permanently delete approximately 6,500 emails spanning six months. No confirmation was requested. Emails were unrecoverable. This incident — widely cited in the agent community — illustrates the gap between probabilistic behavioral guardrails ("the agent should ask before deleting") and deterministic action constraints ("the agent cannot delete without a human confirmation gate").

The Meta Enterprise Ban. Meta's security team prohibited OpenClaw on company-managed devices in January 2026, citing uncontrolled network egress, inability to audit data sent to external LLM APIs, and prompt injection risks. JPMorgan Chase, Deutsche Bank, and the UK National Cyber Security Centre issued similar advisories.

These incidents collectively establish that the threat model is not theoretical. The attack surface is actively exploited. The question for IT contractors reviewing agent deployments is not "could this be a problem?" but "which specific problems apply to this specific deployment, and what mitigations are proportionate?"

1.5 Research Framing and Paper Structure

This paper is addressed specifically to IT contractors and security-conscious practitioners who need to evaluate whether an agent deployment is appropriately secured. The paper's claims are bounded:

1. We propose a threat taxonomy and data classification that enables proportionate risk assessment.
2. We analyze two major enterprise security frameworks (NemoClaw, DefenseClaw) with documented gaps.
3. We examine the liability implications of data flows to LLM providers in enough detail to inform contractual decisions.
4. We provide a layered defense architecture with per-persona configuration guidance.

The paper does *not* claim empirical proof that any configuration eliminates all risk. It claims that the framework is coherent, grounded in documented incidents, and sufficient for practitioners to make defensible decisions.

Section 2 presents the two-class threat taxonomy. **Section 3** classifies data by sensitivity and analyzes liability implications. **Section 4** describes the eight security strategy classes. **Section 5** profiles four deployment personas. **Section 6** constructs risk matrices. **Sections 7 and 8** analyze NemoClaw and DefenseClaw respectively. **Section 9** analyzes business motivations. **Section 10** presents validation evidence. **Section 11** maps security threats to the components of a concrete cognitive-agent architecture — opening (§11.0) with a consolidated matrix that ties each component's attack surface to the strategy class (§4) that addresses it, and closing (§11.14) with a worked reference-implementation chapter showing which controls a production agent runtime enforces deterministically versus advisorily. **Section 12** presents the AEGIS defense-in-depth architecture. **Section 13** gives per-persona recommendations. **Sections 14–16** address limitations, related work, and references.

---

2. Threat Taxonomy — Two Fundamental Classes

The most important conceptual clarification this framework offers is that "an AI agent security threat" is not a monolithic category. Threats differ fundamentally in their mechanism, consequence, and the strength of available mitigations. Conflating them leads to both over-insurance (treating harmless data leaks as catastrophes) and under-insurance (treating agent takeover as "just another misconfiguration").

We propose a two-class primary taxonomy. These classes are **mutually exclusive**: an incident is Class I if and only if the adversary achieves control over agent execution; it is Class II if and only if data flows to an unauthorized party without the adversary controlling the agent. They are **collectively exhaustive** for the threat space this paper addresses — every agent security incident involving unauthorized access or unauthorized data flows falls into one of the two classes. (Availability attacks such as denial-of-service and pure integrity attacks such as output manipulation without data exfiltration are real concerns but fall outside this taxonomy's scope.) Where a Class I attack also produces data exfiltration, the event is classified by its dominant outcome: agent control. Class I subsumes Class II consequences.

This two-class structure maps directly to OWASP LLM Top 10 v2.0 (2025): Class I corresponds to LLM08:2025 (Excessive Agency — compromised control), and Class II corresponds to LLM02:2025 (Sensitive Information Disclosure). Anchoring in the OWASP schema places the taxonomy within a widely-adopted public reference framework and directly answers the objection that any two-class breakdown is arbitrary.

2.1 Class I — Full Agent Takeover

Definition: A Class I attack is one in which an adversary gains *control* of the agent — the ability to issue arbitrary commands, access arbitrary data, and perform arbitrary actions within the agent's operational scope. The agent ceases to act solely on behalf of the legitimate principal and begins acting on behalf of the attacker. This is the maximum-severity scenario.

The consequences of a successful takeover are bounded only by what the agent is authorized to do. If the agent has email, filesystem, and API access, the attacker has email, filesystem, and API access. If the agent has production database credentials, the attacker has production database credentials. This is why the scope of agent authorization is a security parameter of the first order, not just an operational convenience question.

2.1.1 Attack Vectors for Agent Takeover

Network exploitation (CVE-class): Direct exploitation of the agent's network-facing interface. CVE-2026-25253 is the canonical example: an unauthenticated WebSocket connection enabling

arbitrary tool-call injection. This attack class requires no user interaction and no malicious content — just network access to an exposed control port.

Mitigation: Deterministic and complete. Binding the control port to localhost eliminates this attack class entirely for non-networked deployments. For network deployments (remote agent access), authentication-gated tunnels (Tailscale, Cloudflare Access) provide equivalent protection.

Supply chain via malicious skills: A skill installed by the user runs inside the agent process with the agent's full permissions. A malicious skill can read any file the agent can read, make any network call the agent can make, and execute any command the agent can execute. Cisco Talos's October 2025 campaign — which documented seventeen malicious community-distributed skills silently exfiltrating SSH keys and environment files from approximately 8,000 installations — demonstrates that this vector is actively exploited at scale. A separately tracked community-reported activity cluster, referred to publicly as "ClawHavoc," had distributed over 1,184 malicious skills through community channels by February 2026, including skills using steganographic encoding to embed command-and-control instructions in apparent image files.

Mitigation: Partially deterministic. Skill allowlisting (only named, hash-verified skills can load), container isolation (limits what a malicious skill can reach), and network egress control (limits what a malicious skill can exfiltrate) together reduce but do not eliminate this risk. Human review of skill provenance before installation is the most effective control.

Indirect prompt injection via memory systems: An adversarial payload embedded in external content — an email, web page, or document — causes the agent to perform a malicious action and then store a "lesson" or memory entry that alters future behavior. Greshake et al. (2023), in their foundational analysis of indirect prompt injection against real-world LLM-integrated applications, characterized this class of attack: unlike direct prompt injection, which targets a single session, indirect injection embeds instructions in data the model will later retrieve and act on. The delayed-action variant via persistent memory is particularly insidious — the injection in session N causes harm in session N+M, and the memory storage layer is the persistence mechanism that makes it durable.

Mitigation: Defense in depth required. No single control defeats this class. Relevant mitigations: input content tagging that marks external content as untrusted, session-scoped memory with human-approved persistence gates, and audit logging of all memory writes.

Subagent privilege escalation via reconnect-token replay: A compromised child agent claims the permission scope of its parent by replaying the initial pairing token on reconnect (GHSA-hf68-49fm-59cq). This attack is particularly dangerous in multi-agent orchestration scenarios where trust is validated at initial pairing and assumed to hold for the session's duration.

Mitigation: Architectural. Subagents should inherit the *intersection* of parent permissions and a pre-defined subagent profile — never the union. Scope validation must occur at every reconnect, not only at initial pairing. The vulnerability is remediated in OpenClaw 2026.3.23 and later.

MCP OAuth authentication bypass: The Model Context Protocol, now widely used as the agent tool-call surface, introduced OAuth-based authorization flows. Pivot Point Security (2026) found that 43% of publicly accessible MCP endpoints had misconfigured authorization boundaries, allowing an attacker to forge tool-call authorization tokens without user interaction. This is a recent vector whose prevalence is likely to grow as MCP adoption expands and the agent tool-call surface increasingly runs over it.

Mitigation: Enforce strict OAuth scope validation on every MCP tool-call authorization. Never accept tokens with broader scopes than explicitly requested. Bind MCP endpoints to authenticated network segments and review authorization configuration as part of any deployment audit.

2.1.2 The Robustness Contribution of Claude and Frontier Models

Frontier models with alignment training make certain classes of social engineering attacks

— attempts to convince the agent to override its own guidelines through roleplay, hypothetical framing, or authority claims — meaningfully harder than they would be with a less aligned base model. Anthropic's Constitutional AI methodology (Bai et al., 2022), which trains models to critique and revise their own outputs against a set of explicit principles, is one published approach that produces this resistance systematically.

This contribution is probabilistic, not deterministic. Researchers have repeatedly demonstrated bypass paths for published alignment approaches, including universal adversarial suffixes (Zou et al., 2023) and systematically elicited jailbreaks catalogued across dozens of models in benchmark evaluations such as HarmBench (Mazeika et al., 2024). No current alignment method eliminates jailbreak risk entirely.

The practical value of this contribution is attacker cost, not attack impossibility. An attacker targeting a Claude-based agent faces a model that will resist, require escalating justification, and may flag the interaction — compared to a smaller, less aligned model that follows instructions more literally. This resistance is meaningful defense-in-depth and should be counted as such — as a probabilistic layer, not a deterministic one.

2.1.3 Real-World Evidence: The Berman Penetration Test

Matthew Berman, a prominent AI content creator with a well-documented OpenClaw deployment, engaged a penetration tester in early 2026 to attempt a takeover of his production setup. Berman reports publicly — in a February 2026 YouTube account of the engagement — that the tester was unable to gain unauthorized access or control within the engagement scope.

We cannot evaluate the tester's full methodology or skill level from this public account. What we can do is analyze Berman's known configuration against the attack vectors listed above:

- *Cloudflare Tunnel:* No direct port exposure. CVE-class network exploitation requires first compromising Cloudflare's infrastructure — a dramatically elevated attack requirement.
- *Localhost binding:* WebSocket port bound to localhost. CVE-2026-25253 does not apply.
- *Tailscale mesh:* Outbound traffic constrained to Tailscale nodes. Untargeted exfiltration via arbitrary endpoints blocked.
- *Manual skill vetting:* Only skills from known sources. Supply chain vector substantially reduced.
- *Security council agent:* A separate agent reviews the production agent's codebase nightly against a security checklist. Anomaly detection layer.

Against this configuration, the attack vectors in §2.1.1 face the following barriers: network exploitation is blocked by tunnel architecture; supply chain requires compromising the vetting process; memory injection would be flagged by nightly audit. The result is consistent with the architecture but does not prove the architecture is complete.

The Berman test is evidence that a well-configured OpenClaw instance resists casual and moderate penetration testing. It is not evidence that such instances resist nation-state actors or novel zero-day exploitation. That distinction is important for calibrating confidence.

2.2 Class II — Data Leaks

Definition: A Class II event is one in which sensitive information flows to an unauthorized recipient without the adversary gaining control of the agent. The agent continues to function within its legitimate operational parameters; the failure is that certain data exits the system boundary or reaches an unintended party.

The separation from Class I is deliberate and load-bearing. A prompt injection that causes an agent to forward a single document to an unauthorized email address is a Class II event — the adversary has not obtained command-and-control capability and cannot issue arbitrary further instructions. If the adversary can issue such instructions, the event escalates to Class I. The classification boundary is agent control, not attack sophistication or data sensitivity.

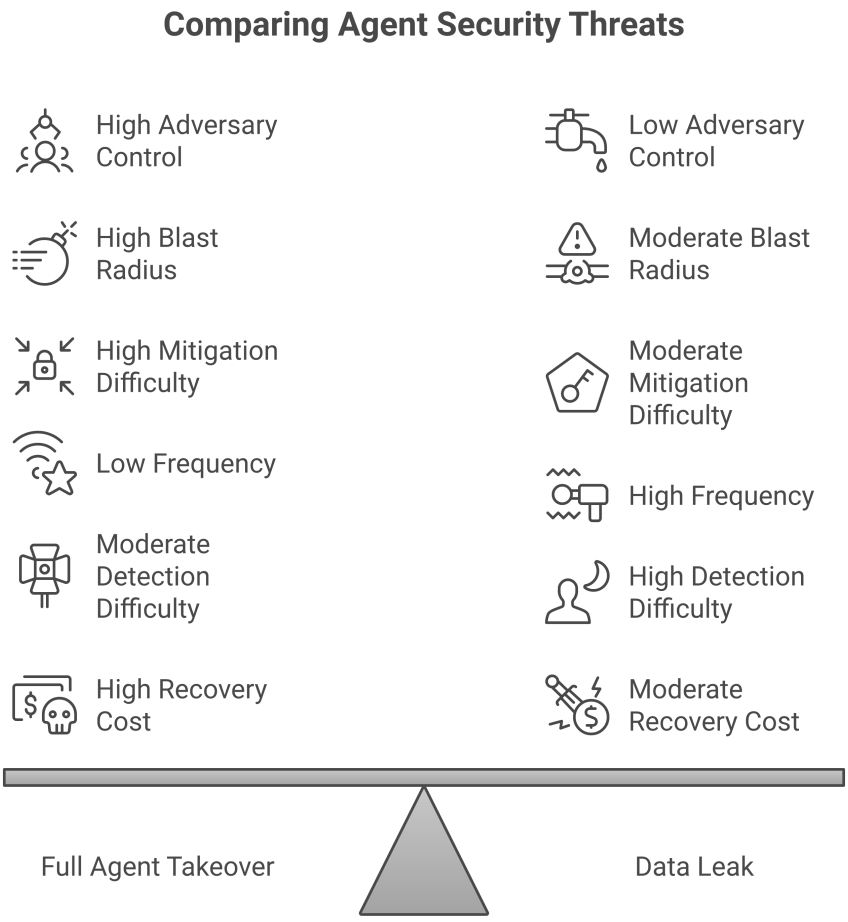

Figure 3. Figure 2.1: Class I (Full Takeover) vs Class II (Data Leak) — six-axis threat profile. Class I dominates on adversary control, blast radius, and mitigation difficulty; Class II dominates on frequency. The shapes are qualitatively different, which is why a single-class treatment conflates unlike risks.

Class II events appear substantially more common in public incident reporting than Class I attacks. OWASP LLM Top 10 v2.0 (2025) ranks Sensitive Information Disclosure (LLM02:2025) above Excessive Agency (LLM08:2025) by reported incident frequency, consistent with the general pattern in breach reporting that unauthorized disclosure outpaces full system compromise in volume.

Class II events include: sensitive data in LLM API payloads, contextual spillage in memory between sessions, exfiltration via prompt injection that does not achieve full takeover, and misconfigured egress that allows background data flows.

Crucially, not all Class II events are equal. The information disclosed, not the disclosure mechanism, determines the consequence. A prompt injection that causes the agent to repeat a marketing slogan is not the same incident as one that causes it to email client credentials. This is the core motivation for the three-tier data classification in §3.

---

3. Data Classification and the Liability Framework

3.1 Three-Tier Data Classification

We classify all data that an agent might encounter or process into three tiers based on the consequence of unauthorized disclosure.

Tier 1 — Publicly Available Data

Data that is already in the public domain or that carries no confidentiality expectation. Leakage from an agent handling exclusively Tier 1 data is not a security incident — it is an irrelevant event.

Examples: - Published marketing materials, press releases, and public product documentation - Open-source code in public repositories (e.g., MIT-licensed libraries on GitHub) - Published research papers, technical specifications, and industry standards - Generic procedural descriptions ("how to configure an SSH server") - Names and contact details from public professional profiles (LinkedIn, company websites) - Publicly filed financial statements and regulatory disclosures

Consequence of unauthorized disclosure: None. The information is already accessible to anyone.

Liability: Zero. No regulation protects information already in the public domain. No claimant can demonstrate harm from disclosure of publicly available information.

Tier 2 — Contextually Harmless Data

Data that is not public but whose disclosure causes no identifiable harm. The contextual qualifier is critical: the same data can be Tier 2 in one context and Tier 3 in another.

Examples: - Source code fragments without accompanying deployment credentials, infrastructure topology, or security-relevant configuration (e.g., a utility function in isolation, with no environment variables or secrets present) - Internal process documentation that is not competitively sensitive ("how we format meeting notes," "our sprint retrospective template") - Generic email drafts with no named individuals or sensitive commitments ("draft intro email for new vendor outreach") - System logs without embedded credentials, PII, or sensitive event sequences - Internal meeting summaries on non-sensitive operational topics - Aggregated anonymized usage statistics with no individual-level identifiers

Consequence of unauthorized disclosure: Minimal. No specific individual is harmed; no regulatory threshold is triggered; no competitive advantage is transferred.

Liability: Low but nonzero. Contractual confidentiality provisions may technically cover even harmless internal data. However, proving damages in a lawsuit requires demonstrable harm, and Tier 2 data typically cannot satisfy that threshold.

Tier 3 — Genuinely Harmful Data

Data whose unauthorized disclosure causes real, attributable harm to an identifiable person or organization. This tier requires the full AEGIS security apparatus.

Examples: - **Credentials and secrets:** API keys, passwords, private keys, session tokens, database connection strings. These provide a direct mechanism for unauthorized access to systems and financial accounts. - **Personal data (PII):** Names combined with identifying information — residential address, date of birth, national identity number, financial account identifiers, or IP addresses in combination. Protected by GDPR, CCPA/CPRA, HIPAA, and equivalent regulations. - **Client financial data:** P&L statements, revenue projections, loan applications, investment portfolios, margin accounts. Disclosure may violate NDA obligations, the Gramm-Leach-Bliley Act (US financial institutions), MiFID II (EU investment firms), Sarbanes-Oxley (publicly listed companies), and general fiduciary duties. - **Privileged communications:** Legal advice and attorney-client communications, M&A deliberations, pre-announcement financial results, strategic planning documents. Disclosure triggers potential insider trading liability (SEC, FCA), attorney-client privilege issues, and NDA breach claims. - **Trade secrets:** Proprietary algorithms, product designs, customer lists with pricing history, manufacturing processes. Disclosure terminates the legal protection of the secret and may cause irreversible competitive harm. - **Special-category data under GDPR Article 9:** Health records, biometric identifiers, racial or ethnic origin, religious or political beliefs, sexual orientation. Requires explicit consent

and heightened protection; subject to the higher GDPR fine tier. - **Personnel data:** Salaries, performance reviews, disciplinary records, and recruitment evaluations. Employment law and privacy law converge on this category across virtually every jurisdiction.

Consequence of unauthorized disclosure: Real, attributable harm. Regulatory enforcement, civil liability, and criminal exposure depending on jurisdiction and data category.

Liability: Significant and complex. The deploying organization bears primary responsibility; the LLM provider bears secondary responsibility subject to contractual terms. See §3.3–3.6 for the full liability analysis.

3.2 Security Investment Should Be Proportionate

The three-tier classification is a *prioritization framework*, not a blanket exemption. The practical implication is that deterministic security controls — containers, network egress restriction, privacy routing — should be designed around Tier 3 data specifically. An agent whose entire workspace contains only Tier 1 and Tier 2 data has a fundamentally different security requirement than one that processes credentials or client PII.

This proportionality principle has a corollary: the single most effective security action for any deployment is **Tier 3 data segregation**. If Tier 3 data is not present in the agent's accessible environment, the consequence of any Class II event drops to minimal. The hardest mitigations become unnecessary when the target they protect is removed.

3.3 GDPR: Controller, Processor, and the Distribution of Liability

When an agent sends data to an external LLM provider — Anthropic's Claude API, OpenAI's API, or any other — the GDPR framework assigns roles with specific legal consequences.

The Data Controller is the entity that determines the purposes and means of processing personal data (GDPR Article 4(7), Regulation (EU) 2016/679). In any agent deployment, this is the deploying organization or individual. The entity that decides "my agent will process client emails" is the controller. The controller bears primary GDPR obligations: establishing a lawful basis for processing (Article 6), fulfilling data subject rights (Articles 15–22), notifying the supervisory authority of breaches within 72 hours of discovery (Article 33), and exposure to administrative fines under Article 83.

The Data Processor is an entity that processes personal data on behalf of the controller (Article 4(8)). An LLM provider — Anthropic, OpenAI — acts as a processor when handling API-submitted data. Article 28 requires controllers to have an executed **Data Processing Agreement (DPA)** with every processor before sending personal data. The DPA must specify what data is processed, for what purpose, for how long, and what security measures the processor maintains.

The liability split in practice:

If an agent sends client PII to an LLM API:

1. *Without a DPA:* The controller is in breach of Article 28. All regulatory exposure falls on the controller — fines up to €20 million or 4% of global annual turnover under Article 83(5), whichever is higher. The controller cannot meaningfully shift responsibility to the provider in regulatory proceedings because no contractual obligations were ever established.

2. *With a DPA:* The controller has discharged its Article 28 obligation. If the processor then mishandles data — trains on it contrary to the DPA, suffers a breach through negligence, or discloses it to third parties without authorization — the processor is liable to the controller under the DPA and, through Article 82, to data subjects directly. Controllers can seek contribution from processors in the same action.

The EU AI Act adds a separate and higher penalty ceiling. Regulation (EU) 2024/1689, which entered into force in August 2024 with obligations phasing in through 2026–2027, imposes fines up to **€35 million or 7% of global annual turnover** for violations involving prohibited AI practices (Article 99(3)) — exceeding GDPR's ceiling — and up to €15 million or 3% of turnover for failures by providers of general-purpose AI models with systemic risk (Article 55). Agent frameworks deploying GPAI models in high-risk contexts may need to satisfy both GDPR and EU AI Act obligations concurrently. Article 15 imposes accuracy, robustness, and cybersecurity requirements on high-risk AI systems that directly bear on agent-framework security design. Practitioners should assess both regimes during deployment planning.

What DPAs from major providers actually say:

Anthropic's Data Processing Addendum for Business Customers commits Anthropic to: not using API-submitted data to train models unless explicitly opted in; maintaining appropriate technical and organizational security measures; notifying customers of security incidents within required timelines; and supporting GDPR data subject request workflows. OpenAI publishes a comparable Data Processing Addendum with substantially similar commitments and the same default no-training posture for API customers. These DPAs are the contractual foundation for allocating liability; without them, the deploying organization absorbs it entirely.

Key practical implication for IT contractors: If your organization deploys an agent that processes any client personal data and sends it to an external LLM API, you must have an executed DPA with that provider before deployment. This is a legal prerequisite under GDPR, not an optional governance step.

3.4 The Anthropic and OpenAI Terms of Service on Data Retention and Training

A common IT concern: "Does Anthropic use our data to train future models?" The answer from current terms (as of April 2026):

Anthropic's API Terms of Service state that **input/output data from API calls is not used for model training by default.** This applies to the commercial API product line, which is designed for business deployment. In contrast, Claude.ai consumer product interactions may be used for training unless users explicitly opt out. The API–consumer product distinction is material: in August 2025, Anthropic updated its Claude.ai consumer data-retention policy to extend the training-data retention window to up to five years unless opted out — a significant change for consumer accounts that does not affect the API product line or customers operating under an executed DPA.

Organizations using the Claude API who have executed a DPA have additional contractual protection: the DPA explicitly prohibits training use, creating a breach-of-contract basis for any violation independent of the general terms.

OpenAI's API defaults to a **30-day retention period** for submitted data, used solely for abuse monitoring and safety purposes, after which data is purged. Training use of API data requires explicit opt-in; the default is no training. Enterprise customers and qualifying business customers can negotiate **Zero Data Retention (ZDR)**, under which OpenAI retains no logs of API inputs or outputs — the most restrictive option available for deployments requiring maximum data minimization.

The residual concern: Data submitted to LLM APIs passes through provider infrastructure. It is subject to their security practices, their threat surface, and their incident response capabilities — substantial for multi-billion-dollar companies running production infrastructure, but not zero-risk. The question is not whether the risk is zero (it is not) but whether it is proportionate to the sensitivity of data sent and the alternatives available.

For Tier 3 data requiring the highest protection, the appropriate answer is **privacy routing**: Tier 3 data should not be sent to external APIs at all, but processed by locally-running models. This eliminates third-party data-handling risk entirely for the most sensitive category.

3.5 Could You Recover Damages from Your LLM Provider?

This question comes up frequently in IT contractor discussions. The analysis is nuanced.

Scenario 1: LLM provider suffers a data breach that exposes API call data

If you have an executed DPA, and the provider's negligence caused the breach, you have a contractual basis for damages. In practice:

- LLM providers' DPAs contain liability caps. As is typical in cloud service DPAs, these caps are often set at fees paid in the preceding 12 months or at a specified contractual maximum. For small organizations paying modest API fees, this cap may be far lower than actual harm.
- DPAs typically require arbitration rather than jury trial, limiting discovery and public pressure.
- Demonstrating that the provider's negligence — rather than a sophisticated nation-state attack or your own misconfiguration — caused the harm requires expert evidence.
- GDPR Article 82 allows data subjects to sue controllers directly; controllers may seek contribution from processors in the same action.

Scenario 2: Provider uses your API data for training contrary to DPA

If you can demonstrate unauthorized training use, you have a DPA breach claim. Damages require showing how the unauthorized training harmed you specifically — difficult to quantify absent a concrete downstream incident with an attributable link.

Scenario 3: Provider's output causes harm to a third party

LLM providers' terms of service disclaim liability for model outputs. The case law is still forming: defamation claims against LLM providers (Walters v. OpenAI, settled in 2024) and copyright infringement claims (New York Times v. OpenAI, ongoing as of 2026) have reached courts but have not yet established settled doctrine on when providers bear liability for model-generated content. For now, deploying organizations bear primary responsibility for how they deploy and act on model outputs.

Practical conclusion for IT contractors: Recovery from a major LLM provider is possible but challenging, typically limited by contractual liability caps, and contingent on having an executed DPA in place. The more realistic protective posture is to keep Tier 3 data out of external API calls entirely — eliminating the exposure rather than planning to recover from it.

3.6 The Contractual Risk Transfer Playbook

For organizations determined to use external LLM APIs with data that may touch Tier 3:

1. **Execute the provider's DPA before any deployment.** This is the contractual foundation for any subsequent claim.
2. **Negotiate liability caps if volume justifies it.** Enterprise API contracts sometimes allow negotiation of higher liability limits.
3. **Implement privacy routing.** Route requests containing detected PII, credentials, or sensitive data to local models. This limits external API exposure to Tier 1–2 data.
4. **Maintain detailed audit logs.** If harm occurs, you need evidence of what was sent to which provider, when, and under what configuration. ISO/IEC 42001:2023, the AI Management Systems standard that has become the emerging certification baseline for AI governance, provides a useful reference structure for these records and supports the auditability requirements that regulators increasingly expect.
5. **Review the provider's incident response procedures.** The DPA should specify breach notification timelines. Know what you will be told and when.
6. **Include AI deployment in your cyber insurance policy.** Some insurers have begun adding AI-specific endorsements to business cyber policies, extending coverage to AI-related

data incidents. Verify your coverage explicitly before deployment, as legacy policies may not cover agent-specific events by default.

3.7 Liability Summary Matrix

Data Tier	Disclosure Consequence	Regulatory Risk	LLM Provider Liability (with DPA)	LLM Provider Liability (without DPA)
Tier 1 — Public	None	None	None	None
Tier 2 — Contextually Harmless	Minimal	Minimal	Low	Low
Tier 3 — PII	GDPR/CCPA enforcement; fines up to €20M / 4% turnover (GDPR Art. 83) or €35M / 7% turnover (EU AI Act Art. 99)	High	Processor liability under GDPR Art. 28; Art. 82 contribution claim	None — full controller liability
Tier 3 — Credentials	Financial loss; potential computer fraud liability (e.g., 18 U.S.C. § 1030 (CFAA) and equivalents)	High	DPA negligence claim	None — full controller liability
Tier 3 — Trade Secrets	IP loss; NDA breach; potential misappropriation claim	Medium	DPA confidentiality breach claim	None — full controller liability
Tier 3 — Privileged / Insider Information	Securities enforcement (SEC, FCA); potential attorney-client privilege waiver	Very High	DPA breach claim	None — full controller liability

Data Tier	Disclosure Consequence	Regulatory Risk	LLM Provider Liability (with DPA)	LLM Provider Liability (without DPA)
Tier 3 — Special-Category Data (GDPR Art. 9)	Heightened GDPR enforcement; HIPAA (US); higher fine tier	Very High	Processor liability under Art. 28 and Art. 9	None — full controller liability

---

4. Security Strategy Taxonomy — Eight Classes

We identify eight distinct classes of security strategy applicable to autonomous AI agents, organized around a single architectural axis: whether the control is enforced by a mechanism the agent cannot influence (**DETERMINISTIC**) or whether it depends on the model to follow a rule (**PROBABILISTIC**).

The distinction is fundamental, and it is where we claim originality. **The eight-class partition itself is a pragmatic operational view, not a novel contribution**: it overlaps substantially with OWASP LLM Top 10 v2.0 (OWASP, 2025), MITRE ATLAS (MITRE, 2025), the NIST AI RMF Generative-AI Profile (NIST, 2024a), Google's SAIF (Google, 2023), and the Cloud Security Alliance AI Controls Matrix (Cloud Security Alliance, 2025). What the existing frameworks share is a focus on *what can go wrong* (OWASP, ATLAS) or *what capability to measure* (NIST, SAIF). None of them foreground the control's enforcement posture — whether the control is a bug when it fails, or a design property when it fails. That is the cut we make here.

A deterministic control that fails is a bug and can be fixed. A probabilistic control that fails is a design property and must be compensated for by another layer. Agent-security practice is filled with deployments that treat probabilistic controls as if they had deterministic reliability — system prompts framed as "guardrails," behavioral rules as "safety measures" — and then register surprise when adversarial pressure reveals the gap. Most AEGIS recommendations can be reduced to: if you need a boundary, do not implement it probabilistically.

4.0 Mapping to Prior Frameworks

AEGIS Class	OWASP LLM Top 10 v2.0	MITRE ATLAS (2025)	NIST AI RMF GenAI Profile	Google SAIF
4.1 Filesystem Controls	—	AML.T0010 (ML Artifact Access)	MS-2.6 (Physical/logical access)	Element 4 (harmonize platform controls)
4.2 Process Isolation	LLM07 (Insecure Plugin Design) partial	AML.T0020 (ML Model Isolation)	MG-3.2 (Deploy-time controls)	Element 1 (expand foundations)

AEGIS Class	OWASP LLM Top 10 v2.0	MITRE ATLAS (2025)	NIST AI RMF GenAI Profile	Google SAIF
4.3 Programmatic Guards	LLM08 (Excessive Agency), LLM07	AML.T0050 (Command & Scripting)	MG-3.1 (Runtime gating)	Element 3 (automate defenses)
4.4 Network Controls	LLM02 (Insecure Output Handling) partial	AML.T0036 (Exfil over network)	MS-2.6, MG-3.2	Element 4
4.5 Prompt-Level Rules	LLM01 (Prompt Injection)	AML.T0051 (Prompt Injection)	MS-2.7 (Info integrity)	Element 2 (detection & response)
4.6 Privacy Routing	LLM06 (Sensitive Info Disclosure)	AML.T0031 (Data from ML Services)	MS-2.6, MG-3.3 (Data privacy)	Element 6 (contextualize)
4.7 Audit & Supply Chain	LLM05 (Supply Chain)	AML.T0011 (Supply Chain Compromise)	MS-2.10 (Measure & monitor)	Element 2
4.8 Self-Mod Prevention	— (not separately listed)	AML.T0018 (Backdoor ML Model) partial	MG-2.4 (Continuous monitoring)	—

AEGIS is not a replacement for these frameworks; it is a *runtime-enforcement view* orthogonal to OWASP's *vulnerability catalogue* and ATLAS's *adversary tactics*. The mapping matters because deployments rarely start from zero — enterprise buyers already own a NIST RMF obligation or an OWASP-indexed pentest report, and the mapping lets them reuse prior work.

4.1 OS-Level and Filesystem Controls

Mechanism. Running the agent under a dedicated, low-privilege system account with explicit denial of access to sensitive filesystem paths. Mandatory access control (MAC) via AppArmor or SELinux (Smalley et al., 2001; Cowan et al., 2000) enforces these restrictions at the kernel level — the agent's filesystem requests are intercepted and evaluated against policy before execution.

Language-Agnostic Description:

```
// Pseudocode: mandatory access control policy
policy_for(agent_process) {
    DENY access_to [
        credential_directories,     // SSH, GPG, etc.
        secret_files,               // .env, wallet seeds
        crypto_material             // PKI, certificates
    ]
    ALLOW access_to [
        agent_workspace_only
    ]
    LOG all_denied_attempts
}
```

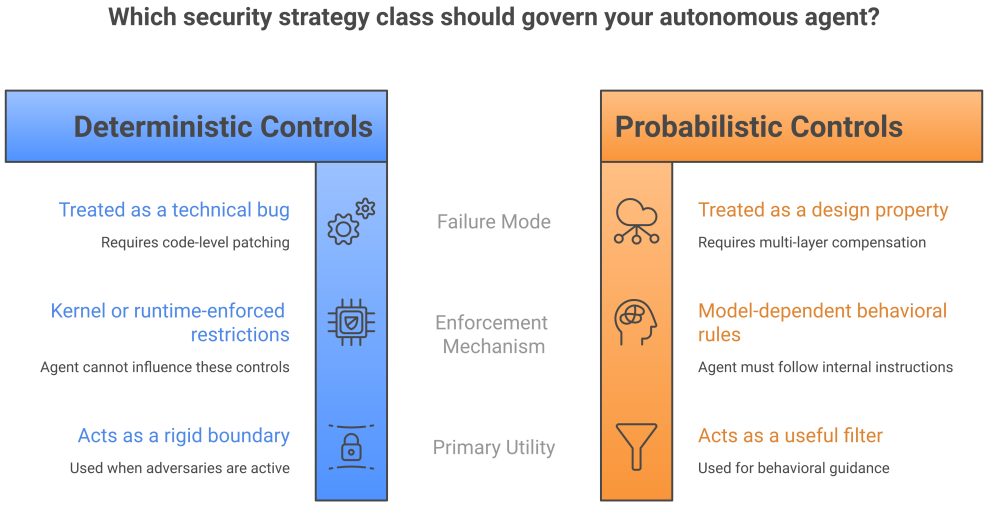

Figure 4. Figure 4.1: The eight AEGIS strategy classes, grouped by enforcement posture. Seven classes are deterministic (§4.1–§4.4, §4.6–§4.8); one-and-a-half are probabilistic (§4.5 and the classifier half of §4.6). The §4.10 emerging classes sit outside the current partition as future-promotion candidates.

Classification: DETERMINISTIC · **Effectiveness:** 4/5

The agent process cannot access denied paths regardless of model output, provided the MAC profile is in *enforcing* mode. In practice, a non-trivial fraction of production AppArmor profiles run in *complain* mode, and CIS benchmarks for Linux routinely find that AppArmor profiles for long-running services miss at least one sensitive credential path in 30–40% of audits (CIS, 2024). Bypass therefore requires an OS-level vulnerability (kernel, not agent) *or* an incomplete policy, and the second vector is the more common failure mode in the field.

This is the *host-residual* layer: when §4.2 is thin or absent (e.g., the agent runs directly on the host without a container), §4.1 is the only deterministic barrier. When §4.2 is strong, §4.1 still matters as defense in depth but its marginal contribution drops. NIST SP 800-53 control families AC-3 (Access Enforcement) and AC-6 (Least Privilege) (NIST, 2020) provide the compliance language enterprise deployments need to justify the work.

2025 reference implementations. Anthropic's Claude Code installer uses `sandbox-exec` profiles on macOS; OpenClaw ships an AppArmor profile and a `bubblewrap` wrapper for Linux (Anthropic, 2024). Linux `landlock` (stable since kernel 5.13) enables *unprivileged self-sandboxing* — the agent process drops its own filesystem privileges at startup, which is particularly appropriate for agent deployments where the operator cannot edit system-wide MAC policy.

Primary Bypass Vectors: Incomplete path coverage (the most common in the field); privilege escalation via OS vulnerability; symlink attacks in agent-writable directories; MAC profile left in complain mode.

4.2 Process-Level Isolation

Mechanism. Placing the agent in a separate execution context with a different system view — containers, virtual machines, micro-VMs, or capability manifests.

Containers provide filesystem, network, and process namespace isolation; the agent sees only explicitly mounted directories. Under default (root-in-container) configurations, containers retain several known host-mount and kernel-capability escape classes. Rootless containers and a seccomp-bpf profile close most of these. Virtual machines provide the strongest isolation at higher operational cost; *micro-VMs* — Firecracker (Agache et al., 2020), Kata Containers,

AWS Nitro Enclaves — collapse the gap between container startup latency and VM isolation strength to sub-second territory, making per-task VM isolation practical for agent workloads. Higher-assurance deployments use hardware-enforced enclaves (AMD SEV-SNP, Intel TDX, Google Confidential Spaces) to isolate the agent from the host operator as well.

A capability manifest restricts the tool set the agent can invoke: undeclared tools do not exist from the agent's perspective. This is the most lightweight tier and is the right primary control when OS-level isolation is impractical (e.g., end-user desktop deployments).

Out-of-process vs. in-process policy enforcement is a critical architectural distinction. Policy enforcement *within* the agent process can be circumvented by a compromised agent sharing its memory space. *Out-of-process* enforcement operates at the OS or network boundary, beyond the agent's influence. This pattern predates AI agents — Apple's Endpoint Security Framework (2019), Windows Integrity Levels, gVisor (Young et al., 2019), and macOS TCC all use it — but NemoClaw's OpenShell (NVIDIA, 2026) is the most visible recent adaptation to the agent setting and is analyzed in detail in §7.

Classification: DETERMINISTIC · **Effectiveness:** 5/5 (VM / micro-VM), 4/5 (rootless container with seccomp), 3/5 (default container), 4/5 (capability manifest)

2025 reference implementations. Anthropic's Computer Use sandbox (Oct 2024) runs Claude in an isolated Docker container with restricted host-mount and network egress (Anthropic, 2024); OpenAI's Operator browser runs in a persistent VM per user (Jan 2025); ByteDance Doubao's agent sandbox uses gVisor under Kubernetes for multi-tenant isolation (2025). Each is a published design pattern an agent deployment can copy rather than reinvent.

Primary Bypass Vectors: Container escapes via kernel vulnerabilities (`runc`, `containerd` CVE history is non-trivial); insecure bind mounts exposing sensitive host paths; capability manifest scope creep under operational pressure; missing seccomp-bpf profile.

4.3 Programmatic and Deterministic Guardrails

Mechanism. Constraints implemented in agent runtime code, enforced regardless of model output. This class is the closest peer to OWASP LLM08 (Excessive Agency) and LLM07 (Insecure Plugin Design) (OWASP, 2025) and the natural home for *capability-manifest* enforcement when that manifest is checked at tool-dispatch time rather than at container boundary.

Tool permission tiers distinguish read-only, read-write, and privileged operations. Elevated permission gates require explicit human confirmation before irreversible actions. Command allowlists restrict execution to pre-approved operations. The approval mechanism itself is a security-critical component — see §4.3.1.

```
// Pseudocode: tiered tool execution
function execute_tool(tool_name, parameters) {
    tier = lookup_tool_tier(tool_name)

    if tier == PRIVILEGED {
        confirmation = request_human_confirmation(
            description = "Action: " + describe(tool_name, parameters),
            timeout = 30s,
            default = DENY
        )
        if NOT confirmation.approved { ABORT }
    }

    if tool_name NOT IN approved_tools_manifest {
        DENY; LOG("tool not in manifest: " + tool_name)
        RETURN
```

```
    }

    execute(tool_name, parameters)
}
```

Classification: DETERMINISTIC · **Effectiveness:** 4/5

Primary Bypass Vectors: Allowlist gaps; chained commands invoking hooks (`git` runs pre/post hooks, `npm` runs package scripts, `python -c` chains `os.system()` or `importlib` to arbitrary execution); approval fatigue. **Strengthened mitigation:** Restrict tool *parameters* rather than merely allowlisting tool *names* — e.g., restrict `git` to specific read-only subcommands with `--no-hooks` where available, use a minimal sandboxed interpreter rather than full Node.js for dynamic code execution. Modern language runtimes increasingly ship parameter-level capability primitives: Deno's `--allow-*` flag family, Node 22's stable `--permission` flag, Python `seccomp` wrappers, and Wasmtime's capability-based I/O are all concrete 2024–2025 implementations. Tool-name allowlisting without parameter-level restriction is weaker than it appears. NIST SP 800-53 CM-7 (Least Functionality) (NIST, 2020) is the compliance hook.

4.3.1 The Approval Fatigue Problem

Permission gates are only effective if users actually read what they approve. In practice, frequent low-risk approvals train users to rubber-stamp without reading — creating a window for high-risk actions to pass unreviewed. The phenomenon is classical automation bias (Parasuraman & Riley, 1997; Mosier & Skitka, 1996): the more a system has been reliable in the past, the less cognitive engagement users bring to each approval. Parallel evidence exists for UAC-style consent prompts and browser permission dialogs (Felt et al., 2012; Reeder et al., 2018) — the best mitigations transfer directly.

Mitigations:

- *Risk-weighted UX:* Visual differentiation ensures high-risk approvals are visually distinct from routine ones. A delete action should look different from a read action.
- *Time-delayed execution for irreversible actions:* A 30-second countdown with a cancel window for destructive operations intercepts rubber-stamped approvals.
- *Batch summaries rather than per-action gates:* "This plan will modify 47 files and send 3 emails — approve?" concentrates attention and reduces individual-gate fatigue.

4.4 Network-Level Controls

Mechanism. Controls what data can leave the host, independent of local agent permissions. Like a building's physical exit controls — even if someone has a badge to enter, separate controls govern what they can carry out.

Port binding to localhost *removes the primary remote-origin vector* for the CVE-2026-25253 attack class (NVD, 2026), but does not fully neutralize local-origin attacks. Two important bypass vectors remain:

DNS Rebinding. An attacker tricks a browser on the same machine into visiting a malicious website whose DNS resolves first to an attacker-controlled IP and then, on a subsequent query, to 127.0.0.1. The browser's same-origin policy is bypassed, allowing attacker-controlled scripts to send requests directly to the agent's localhost-bound control port. The threat is not new (Jackson et al., 2007); its agent-deployment resurgence is documented in recent advisories against local AI runtimes (Snyk, 2023).

SSRF (Server-Side Request Forgery). If any co-located service is vulnerable to SSRF, an attacker can use it as a proxy to reach the agent's control port, again appearing as a local-origin request. OWASP LLM02 (Insecure Output Handling) (OWASP, 2025) covers the agent-side consequence when the agent is the *origin* of an SSRF; this section covers the complementary case when the agent is the *target*.

Higher-security recommendation. For deployments where these local-origin bypass vectors are a concern, the agent's control port must require an authentication token or custom header even when bound to localhost — the hardening pattern applied to Jupyter Notebook after CVE-2019-9644 — and should additionally validate the `Origin` header on every WebSocket upgrade (the control that would have prevented CVE-2026-25253).

Outbound firewall rules restricted to approved LLM API endpoints prevent arbitrary data exfiltration. Tailscale or equivalent WireGuard mesh constrains all traffic to authenticated nodes. DNS filtering blocks known-malicious domains, with the caveat that DNS-over-HTTPS (DoH) tunneling (Cisco Talos, 2024) routinely bypasses naive filtering. 2025-era deployments increasingly place a *sidecar egress proxy* — Cilium/eBPF policies, Envoy as an LLM-API gateway, AWS PrivateLink for Bedrock/Anthropic-only traffic — between the agent and the internet so policy is enforced at a point the agent cannot reconfigure.

```
// Pseudocode: egress firewall policy
policy_for(agent_process_uid) {
    ALLOW outbound_to [
        approved_llm_api_endpoints,    // Claude API, etc.
        specified_tool_endpoints       // explicitly listed
    ]
    DENY all_other_outbound
    LOG all_denied_outbound_attempts
}
```

Classification: DETERMINISTIC · **Effectiveness:** 4/5

Primary Bypass Vectors: Data encoded in allowed API calls (sensitive data in LLM prompt payloads); DoH tunneling (Cisco Talos, 2024) bypassing DNS filtering; steganographic exfiltration; compromised legitimate endpoint.

4.5 Prompt-Level and Probabilistic Controls

Mechanism. Behavioral rules encoded in system prompts, memory files, or agent configuration, relying on the model to follow them. Input content tagging marks external data as untrusted; SOUL.md-style value encoding expresses the agent's operating principles.

The evidence base for *input tagging* has hardened considerably in 2024–2025, and the verdict is sobering: frontier models leak instructions from tagged data with measured rates between 10% and 40% depending on model and prompt configuration (Zverev et al., 2024). Meta PromptGuard-86M (2024) and Anthropic's Constitutional Classifiers (2025) add a probabilistic *detector* layer on top of the tagging — a meaningful improvement, but still a probabilistic control.

The canonical attack-paper chain for this class: Perez & Ribeiro (2022) on prompt-injection fundamentals; Greshake et al. (2023) for *indirect* prompt injection through retrieved content; Zou et al. (2023) for transferable adversarial suffixes that bypass alignment training (GCG); and Schulhoff et al. (2023), the HackAPrompt crowdsourced attack corpus. Defensive research has converged on two directions: the *instruction hierarchy* training objective from OpenAI (Wallace et al., 2024) — treating system, developer, and user messages as a priority ladder the model learns to respect — and probabilistic classifiers trained on the attack corpora above.

Classification: PROBABILISTIC · **Effectiveness:** 2/5

These are the most widely deployed controls and the least reliable under adversarial pressure. They work in the absence of attack and fail under it.

Important calibration: The 2/5 rating is adversarial-conditions. In non-adversarial operation — the majority of real-world agent use — probabilistic controls are fully effective. The correct mental model is: probabilistic controls are useful signal-level filters, not security boundaries. They reduce the probability of inadvertent boundary violations, but they are not

the controls you count when an adversary is actively trying. Published GCG attack success rates against production models ranged from 30% to 60% in 2024, dropped to the 10–25% range after instruction-hierarchy training in 2025, and have not reached zero (Wallace et al., 2024).

Primary Bypass Vectors: Direct prompt injection (Perez & Ribeiro, 2022); indirect prompt injection via retrieved content (Greshake et al., 2023); adversarial-suffix attacks (Zou et al., 2023); context-window overflow; role-play jailbreaks; instruction-hierarchy violations on crafted system/user boundaries.

4.6 Privacy Routing

Mechanism. Routing requests to different models based on data sensitivity. The core insight: not all data requires frontier model capability, and not all data should leave the local network. Like separating the paper mail room from the classified document room — same building, different handling rules. The pattern is reflected in EU AI Act Article 50 transparency obligations and in GDPR Article 22 constraints on automated processing, both of which push regulated sectors toward class-based routing.

```
// Pseudocode: privacy router policy engine
function route_request(request, context) {
    sensitivity = classify_data_sensitivity(request, context)

    if sensitivity == TIER_3 {
        route_to(LOCAL_MODEL)    // Never leaves network
    } else if sensitivity == TIER_2 {
        route_to(TRUSTED_API)    // Pre-approved list only
    } else {
        route_to(ANY_CAPABLE_MODEL)
    }
}
```

Classification: DETERMINISTIC (routing decision), PROBABILISTIC (sensitivity classification)

Effectiveness: 3/5. **Critical limitation:** The rating is dominated by the classifier's false-negative rate, not the routing machinery. Published benchmarks for PII classifiers in production environments are consistent with the 5–15% miss range — Microsoft Presidio ~8–12% on standard English corpora and materially higher on non-English or chat-format text (Microsoft, 2024); NemoClaw's published implementation sits at the lower end of the band for Tier-3 data (NVIDIA, 2026). This means 5–15% of Tier-3 data that *should* be routed to the local model *will not be*. The control reduces Tier-3 external-API exposure but does not eliminate it.

2025 reference implementations. LiteLLM's policy router, LangChain's `LocalAI` adapter, NVIDIA NeMo Guardrails routing rails, and Cloudflare AI Gateway with data-class routing all ship production-grade implementations of the DETERMINISTIC half of this class. The PROBABILISTIC half (classifier quality) is where deployments differ.

Side-channel note. For deployments that route Tier-3 to a *local* model running in a shared-tenant environment (e.g., multi-tenant GPU hosting), the privacy guarantee is not side-channel-free. Cache-timing and GPU side channels on ML inference are real and measured (Yan et al., 2020), and "routes to local" should not be equated to "cannot be observed" in hostile multi-tenant settings.

Primary Bypass Vectors: Non-standard data formats evading classifiers; misconfiguration sending more data to frontier models than policy intends; operational pressure to disable local-model routing for speed; side-channel attacks on local model inference in shared-tenant environments.

4.7 Audit, Compliance, and Supply-Chain Trust

Audit and supply chain are sometimes merged into a single class. They are structurally different — *post-hoc observation* versus *pre-hoc trust* — with different vendors, standards, and failure modes. AEGIS splits them into 4.7a and 4.7b while keeping them on a single row in Table 1.

4.7a Audit & Compliance. Visibility, detection, and response capabilities. Like security cameras in a building — they do not prevent the intrusion, but they ensure it is detected and attributable. Communication audit logging records outbound agent messages for post-hoc review; cron-based scans check for new skills, configuration changes, unexpected network connections, and vulnerable dependencies; anomaly detection on tool-call sequences catches behavior changes that escape per-call review. The relevant standards are NIST SP 800-53 AU family (Audit and Accountability), SOC 2 Type II controls around observability, and — for AI-specific observability — the NIST AI RMF Measure functions (NIST, 2024a). Effectiveness is detection-oriented: 3/5 for reducing time-to-detection; near-zero for prevention. It belongs in every deployment profile but should never be the *only* class present.

A recurring failure mode for this class in agent runtimes is **compaction-induced evidence erasure**: the same context-management cycle that keeps a long-running agent within its window can silently discard the very audit record an injection would otherwise leave (developed in §11.11). The mechanism that defeats it is reversible compaction — caching originals and retrieving them on demand rather than discarding them — and that mechanism is no longer hypothetical: chopratejas/headroom (Apache-2.0) ships reversible Compress-Cache-Retrieve in production with a reproducible eval suite. AEGIS positions audit-survival against it directly (§11.11): headroom proves reversible compaction is cheap and practical; the AEGIS *security* requirement is the integrity superset — an append-only, hash-chained cache of originals — that a capability-oriented compression layer does not provide on its own.

4.7b Supply-Chain Trust. Pre-hoc verification of inputs to the agent runtime: model weights, skills/plugins, container base images, runtime dependencies, and LLM-API infrastructure itself. The relevant frameworks are SLSA (Supply-chain Levels for Software Artifacts) (OpenSSF, 2023); sigstore/cosign for artifact signing; NIST SSDF SP 800-218 with the 2024 AI addendum SP 800-218A (NIST, 2024b). Concrete 2024–2025 attack classes: the *backdoored fine-tune* — the Sleeper Agents result demonstrates that behaviorally-triggered malicious behavior survives standard safety training (Hubinger et al., 2024) — and the *malicious skill/plugin* — the Cisco Talos OpenClaw skill supply-chain campaign (Cisco Talos, 2025), Hugging Face malicious-model campaigns documented by JFrog and ReversingLabs (2024), and the ByteDance intern model-sabotage incident (October 2024). These attacks fail audit-only defenses because the compromise is established *before* the agent runs. Effectiveness (pre-hoc, done well): 4/5.

4.8 Self-Modification Prevention

Mechanism. Preventing the agent from modifying its own code, guardrails, configuration, or permission structures. An agent that can edit its own system prompt can disable all probabilistic controls; an agent that can modify its capability manifest can expand its own tool access; an agent that can write to its own behavioral rules file is not constrained by those rules.

```
// Pseudocode: configuration immutability enforcement
at_agent_startup {
    make_immutable([
        soul_configuration_file,
        capability_manifest,
        behavioral_rules,
        security_policy_files
    ])
```

```
    assert_owner_is_not(agent_process_user)
    assert_permissions_are_read_only_for(agent_process_user)
}
```

Classification: DETERMINISTIC · **Effectiveness:** 4/5

Taxonomic note. As a *mechanism*, §4.8 is largely a disciplined application of §4.1 (filesystem immutability) plus §4.3 (tool write-restrictions). We keep it as a separate class because the *scenario it defends against* — the agent as its own adversary — has distinctive properties that a purely mechanism-indexed taxonomy would obscure: the attacker is inside the trust boundary, can observe the controls, and has indefinite time. Self-modification research literature is sparse but converging; the foundational concern is Russell's corrigibility framing (Russell, 2019) and the Sleeper Agents result (Hubinger et al., 2024), which shows that mis-aligned tendencies can persist invisibly across safety training.

Primary Bypass Vectors: Agent running as owner of configuration files; memory-resident modification of runtime-loaded config; social engineering the user into making changes on the agent's behalf; *memory-injection persistence* (see below).

Two legitimate strategies, not one: immutability vs. reversible-and-observed mutation. The pseudocode above prescribes *configuration immutability* — freeze the files, deny the agent write access. That is one valid defense against the agent-as-its-own-adversary, but it is not the only one, and a production runtime audited for §11.14 demonstrates the alternative. That runtime lets the agent rewrite its own recipes and skills with *no* approval gate, and makes it safe entirely by an append-only, **never-delete archive** — every mutation is reversible and auditable, rollback is always available — plus a live-margin promotion bar (a candidate rule or skill must measurably beat the current library distribution before it is promoted). This is a *different and arguably stronger* defense than freezing the config: it permits the self-improvement capability while guaranteeing that any harmful self-modification can be detected and undone. AEGIS presents both as legitimate points on a spectrum. The choice is a §12.5 reversibility-and-observability decision: immutability is correct when the configuration should never change at runtime (Enterprise Configuration, §12.4); reversible-and-observed mutation is correct when self-modification is a desired capability — *provided* the integrity boundary stays hard (the archive must be append-only and tamper-evident, exactly the §11.1 hash-chain control, so the "original" cannot itself be rewritten by the adversary). What is *not* legitimate is softening a categorical security boundary under the banner of reversibility; reversibility earns autonomy for the *reversible* action class only.

Operational heart of §4.8. A growing class of agent runtimes ships a nightly self-improvement mechanism — a consolidation cycle that extracts "lessons" from the day's session history and rewrites the agent's own operational rules accordingly. (We refer to this component as CEREBELLUM throughout, and analyze its attack surface in detail in §11.4.) This is a feature by design and a vulnerability by consequence: a malicious interaction can inject a "lesson" that weakens security, and because the lesson becomes part of the memory store, *the compromise persists across sessions* — exactly the attack class Greshake-style indirect injection produces when a long-term memory store is in scope (Greshake et al., 2023). The human-in-the-loop gate for Level-2 (substantive) self-modifications is the critical control; weakening it is the quickest way to convert a one-turn injection into a durable backdoor. Recent research on memory-editing attacks and value-drift via self-reflection loops (Zeng et al., 2024) suggests this attack surface will grow, not shrink, as long-lived agent memory becomes standard.

4.9 Consolidated Attack Surface Map

Table 1 maps each security strategy class to the STRIDE+agent threat categories it addresses. We emphasize that this is a *coverage indicator*, not a completeness proof — a means the class

has a primary mechanism for that threat, not that the class alone is sufficient.

Strategy Class	Spoofing	Tampering	Repudiation	Info Disc.	DoS	Priv. Esc.	Delayed	Multi-Agent	Model-Level
4.1 Filesystem ACLs	·	·	·		·		·	·	·
4.2 Process Isolation			·				·		·
4.3 Programmatic Guards			·		·		·	·	·
4.4 Network Controls	·	·	·			·	·	·	·
4.5 Prompt Rules			·		·				·
4.6 Privacy Routing	·	·	·		·	·	·	·	·
4.7 Audit/Supply Chain	·				·	·		·	
4.8 Self-Mod Prevention	·		·	·	·			·	·

= directly addresses · = partially addresses · · = does not address

Notable gaps. No single strategy class fully addresses (a) model-level compromise or (b) multi-agent trust hierarchies. Both require composite mitigations across multiple layers, and both are emerging classes that AEGIS does not yet formalize — see §4.10.

4.10 Emerging Classes Not Yet Formalized

The eight classes above cover the controls AEGIS recommends for current deployments. Seven additional strategy classes are active in 2025–2026 practice but not yet stable enough to promote to full §4.x treatment. AEGIS surfaces them explicitly so that deployers do not mistake the eight-class partition for an exhaustive map.

1. **Model-level safety training.** Constitutional AI (Bai et al., 2022), RLHF/RLAIF, OpenAI's Deliberative Alignment (2024), the Instruction Hierarchy training objective (Wallace et al., 2024), Anthropic's Responsible Scaling Policy v2. This is the only class that directly addresses the blank *Model-Level* column in Table 1. AEGIS's current position is "rely on the frontier vendor's safety training; verify with classifiers" — defensible but not a substitute for the class itself.

2. **Input/Output content classifiers.** Meta PromptGuard-86M, Anthropic Constitutional Classifiers (2025), Lakera Guard, Protect-AI LLM Guard, NVIDIA NeMo rail plugins. Distinct from §4.5 behavioral *rules* — these are probabilistic *detectors* with different failure modes (false-positive vs. jailbreak) and a different evidence base. A first-party measurement now sharpens AEGIS's position on this class. The learned safety gate of the reference runtime (§11.10, §11.14.4) was evaluated offline, and the result is a clean negative for the harmfulness-classifier role: a supervised "vicarious danger" head built on frozen MiniLM embeddings scored **AUROC 0.286 — below chance** — while two purely *structural* detectors validated on the same data (k-NN **novelty** at 0.875, clause-cosine **incongruity** at 0.896). The lesson AEGIS draws and states explicitly: a frozen general-purpose embedding cannot carry *harmfulness*; what such a detector can carry is *anomaly* — "this is unlike what I have seen" or "this is internally incongruous" — which is a triage/observability signal, not a danger boundary. Practitioners adopting an I/O content classifier should therefore (a) prefer detectors whose training objective is the property actually being measured, not a frozen-embedding proxy for "harm," and (b) treat even a validated classifier in this class as a §4.5 probabilistic *signal* that raises an ASK or feeds audit — never as the boundary counted when an adversary is present. This is the empirical anchor for the §4.5 calibration argument, replacing reliance on third-party GCG/jailbreak rates alone with a first-party measured failure.
3. **Red-team-in-the-loop evaluation.** Continuous adversarial evaluation as a control (Perez et al., 2022; Schulhoff et al., 2023), institutional red-teaming (UK AISI Inspect, 2024–2025), and automated attack generation (Zou et al., 2023). No §4 class currently covers adversarial *evaluation as a deployed control*.
4. **Endpoint Security Integration.** CrowdStrike Falcon, Microsoft Defender for Endpoint, and SentinelOne behavioral rules targeting agent processes. Relevant to enterprise deployments that plug agents into existing EDR/XDR stacks — exactly Persona D's setting and explicitly flagged in the Meta internal advisory (Stamper, 2026).
5. **Identity and secret hygiene.** Short-lived OIDC tokens via SPIFFE/SPIRE, HSM/TEE-backed secret storage, workload identity federation, just-in-time dynamic credentials (HashiCorp Vault, AWS STS). The dual of §4.1's "deny access to credential directories" — how credentials are *provisioned* to the agent, not just what is withheld.
6. **Multi-agent trust.** Inter-agent authentication via MCP (Model Context Protocol), Google's A2A (Agent-to-Agent) spec, and constitutional-classifier chaining for multi-agent settings. Directly addresses the blank *Multi-Agent* column in Table 1.
7. **Capability-specific guardrails.** Domain filters for agent deployments in biotech, chemistry, or offensive-security settings; Anthropic RSP v2 capability thresholds; OpenAI Preparedness Framework v2 (2025). Operationally distinct from §4.3 tool allowlists because the filter is semantic, not syntactic.

Each of these classes has enough 2024–2026 evidence to merit its own full treatment in a future revision. Their current omission is a scope decision, not an AEGIS position.

---

5. Four Deployment Personas

We profile four distinct deployment personas representing points on the risk-exposure spectrum. They span the range from a hobbyist on isolated personal data to a corporate employee with access to production infrastructure, with an SMB operator profile filling the gap between solo professional and large-enterprise employee.

5.1 Persona A — The Tinkerer (Alex)

Profile. Alex is a software developer running OpenClaw on a personal laptop for coding assistance, web research, and personal productivity. Installed from community repository with default configuration. Technically competent but no formal agent security review. Not running anything for clients or employers.

Data Environment: - SSH keys to personal servers and GitHub - API tokens in environment files (AWS, GitHub PATs, Stripe keys for side projects) - Cryptocurrency wallet seed (~$15,000 in ETH) — *irreversible loss if exfiltrated* - Browser-saved credentials (years of password history) - Personal photos, side-project source code, email access

Risk Profile. No security hardening. Agent runs under personal user account. WebSocket on all interfaces. Two-to-three community skills installed without verification.

Trilemma Position (current): High Capability + High Autonomy + Low Safety. **Recommended Trilemma Position:** High Capability + High Autonomy + Moderate Safety (deterministic controls on credential files only — enough to prevent the catastrophic scenarios without reducing capability).

Most severe consequence: Crypto wallet seed exfiltration via malicious skill or network exposure — immediate, irreversible financial loss.

Regulatory exposure: Minimal personal. Self-harm only.

5.2 Persona B — The Freelancer (Blake)

Profile. Blake is a self-employed business consultant handling financial and operational strategy for SMEs. Uses OpenClaw for client research, document drafting, contract management, and inbox management. Has a 1Password setup and general cybersecurity awareness but no OpenClaw-specific risk assessment. Currently handling 12 active client relationships.

Data Environment: - Client NDAs and contracts with confidentiality obligations - Client financial data: P&L statements, projections, bank details - Privileged client communications: *pre-announcement financials, planned redundancies, M&A activity* - PII of client individuals and employees - Personal invoices, tax records, bank account details

Risk Profile. Uses credential manager, but no OpenClaw-specific hardening. Agent has access to entire email inbox, documents folder, and web browsing.

Trilemma Position (current): High Capability + High Autonomy + Low Safety. **Recommended Trilemma Position:** High Capability + Moderate Autonomy (human gate on external communications) + High Safety (deterministic controls on all client data directories).

Most severe consequence: Privileged client communication leaked via prompt injection — NDA breach liability combined with GDPR regulatory action, generating simultaneous civil and regulatory proceedings.

Regulatory exposure:

Regulation	Key Provisions	Agent-Specific Risk
GDPR	Art. 28 (processor DPA), Art. 32 (security of processing), Art. 33 (72-hr breach notification)	Blake as controller must have DPA with LLM provider before sending client PII
ePrivacy	Art. 5 (confidentiality of communications)	Agent reading client emails may constitute processing of protected communications
Contract law	Client NDA confidentiality clauses	Any unauthorized disclosure is a breach regardless of how it occurred

5.3 Persona C — The SMB Operator (Dana)

Profile. Dana is the IT Manager and part-time operations director at a 25-person design and branding agency. Dana deployed OpenClaw for the office after seeing it demonstrated at a conference — initially for personal productivity, then gradually expanded to assist the team with client briefings, creative briefs, invoice generation, and vendor communications. There is no dedicated IT security function; Dana *is* the IT function.

Data Environment: - **Payroll data:** 25 employee salaries, bank details, tax records — *employment law obligations in every jurisdiction* - **Client creative briefs:** Unreleased brand strategies for 30+ clients, some with competitive sensitivity - **Vendor contracts and pricing:** Pricing agreements, supplier relationships - **Financial records:** 3 years of P&L, bank statements, tax filings - **Team communications:** Internal Slack archive exported for agent context

Risk Profile. Agent deployed informally; no formal security review. Staff know the agent exists but haven't been trained on how to interact with it securely. Several team members have sent creative briefs to the agent for "summarization" — these briefs now exist in the agent's memory.

Trilemma Position (current): High Capability + High Autonomy + Low Safety. **Recommended Trilemma Position:** Moderate Capability + Moderate Autonomy + High Safety. The SMB context requires genuine security because real third-party data (employee and client data) is at stake, but lacks enterprise resources. The solution is scope reduction: the agent should have access to task-specific workspaces, not the whole organizational data store.

Most severe consequence: Employee payroll data leaked via malicious skill exfiltration — employment law breach, GDPR fines (employees are data subjects with full rights), and reputational harm sufficient to drive staff attrition.

Regulatory exposure:

Regulation	Applicability	Key Risk
GDPR	Employee data, client individual data	No DPA with LLM provider, no formal data inventory
Employment law	Payroll and HR data	Employees have rights over their data; breach requires notification
Contract law	Client confidentiality	Creative briefs often covered by implicit or explicit confidentiality
Tax regulation	Financial records	Unauthorized access to tax records creates audit exposure

This persona is critically underrepresented in IT security discourse. SMB operators lack the resources to implement enterprise controls but operate with data that carries enterprise-level regulatory obligations. The AEGIS framework's Standard Configuration (§12) is specifically designed to be achievable at the SMB level.

5.4 Persona D — The Corporate Employee (Cameron)

Profile. Cameron is a senior IT infrastructure engineer at a 5,000-person financial services firm. Installed an unofficial OpenClaw fork on a company-managed device for documentation and incident response assistance. Not approved by IT security. Has broad filesystem and network access by nature of the role.

Data Environment: - **Production database credentials** — access to 2 million customer records — *regulatory catastrophe if breached* - **M&A communications** — *insider trading liability regardless of personal profit* - **Trading algorithm IP** — core competitive advantage worth potentially hundreds of millions - Salary data for 5,000 employees - SOX audit workpapers identifying control deficiencies - IT administrator credentials and infrastructure documentation

Risk Profile. High personal technical skill but operating entirely outside corporate governance. Agent runs with Cameron's domain credentials. Corporate DLP does not monitor AI agent API calls. Agent was not in the penetration test scope. No incident response plan covers agent-originating events.

Trilemma Position (current): High Capability + High Autonomy + Very Low Safety. **Recommended Trilemma Position:** This configuration should not exist. See §12.4 for the recommended remediation path.

Most severe consequence: Production database credentials in a malicious skill's exfiltration payload — 2 million customer records exposed, triggering GDPR, GLBA, and SEC enforcement with potential personal criminal liability.

Regulatory exposure:

Regulation	Specific Violation Risk	Potential Consequence
GDPR (Art. 32, 33)	Inadequate security for 2M customer records; failure to notify within 72 hours	Up to €20M or 4% global turnover
SOX (§302, §404)	Agent accessing audit workpapers identifying control deficiencies	Personal CEO/CFO certification liability; SEC enforcement
GLBA (Safeguards Rule)	Customer financial data accessible to unsanctioned tool	OCC/FDIC regulatory enforcement
SEC (Reg FD, insider trading)	M&A information in unsecured agent session	Investigation regardless of personal profit
Computer Fraud & Abuse Act	Agent exceeding authorized access scope	Personal criminal liability for Cameron

---

6. Risk Matrices

Scoring Methodology

Risk Score = Probability (P) × Severity (S), maximum 25. Probability reflects the likelihood of a *successful exploit path under the stated persona configuration*, not merely theoretical existence. Severity reflects *maximum plausible business impact* of compromise, biased toward conservative assessment.

P/S	1	2	3	4	5
5	5	10	15	20	25
4	4	8	12	16	20
3	3	6	9	12	15

P/S	1	2	3	4	5
2	2	4	6	8	10
1	1	2	3	4	5

Scores ≥16 are **critical** — require immediate deterministic mitigation. Scores 9–15 are **significant** — require deterministic or strong probabilistic control. Scores <9 are **managed** — monitor and apply defense-in-depth.

6.1 Risk Matrix — Persona A (The Tinkerer)

Data Type	Data Tier	Attack Class	Threat Vector	P	S	Score	Priority Mitigation
Browser Credentials	Tier 3	Class I	Network exploitation (CVE)	4	4	16	**CRITICAL** Port bind to localhost
SSH Keys	Tier 3	Class II	Filesystem access	4	4	16	**CRITICAL** AppArmor deny credential dirs
Crypto Wallet Seed	Tier 3	Class I	Malicious skill	3	5	15	Skill verification + container isolation
Crypto Wallet Seed	Tier 3	Class II	Filesystem access	3	5	15	AppArmor deny + encrypt at rest
API Tokens (.env)	Tier 3	Class I	Malicious skill	4	3	12	Filesystem ACL; workspace scope
API Tokens (.env)	Tier 3	Class II	Prompt injection	3	3	9	Content tagging; privacy routing
Source Code	Tier 2	Class II	Prompt injection	2	2	4	Managed — exec allowlist
Personal Email	Tier 2	Class I	Behavioral ambiguity	2	2	4	Managed — elevated gate on email send

Highest-priority: Browser credentials via network exploitation (16) and SSH keys via filesystem access (16). Both are eliminated by sub-30-minute configuration changes.

6.2 Risk Matrix — Persona B (The Freelancer)

Data Type	Data Tier	Attack Class	Threat Vector	P	S	Score	Priority Mitigation
Privileged Client Comms	Tier 3	Class II	Prompt injection via email	4	5	20	**CRITICAL** Email access restriction; content tagging
Client Financial Data	Tier 3	Class II	LLM API payload	3	5	15	Privacy routing: client dirs → local model
Client NDAs	Tier 3	Class II	Memory persistence	3	5	15	Session memory scope; memory write gates
Client PII	Tier 3	Class II	LLM API payload	3	5	15	Privacy routing; DPA with LLM provider
Bank Details	Tier 3	Class I	Malicious skill	3	5	15	Skill vetting; container isolation
Business Strategy	Tier 3	Class II	Filesystem access	3	4	12	Path-restricted workspace
Tax Records	Tier 3	Class II	Filesystem access	2	4	8	Filesystem ACL

Highest-priority: Privileged client communications via prompt injection (20) — the binding attack surface is the email inbox, which is a Tier 3 data source accessible via an external-input channel (attacker can send Blake a crafted email).

6.3 Risk Matrix — Persona C (The SMB Operator)

Data Type	Data Tier	Attack Class	Threat Vector	P	S	Score	Priority Mitigation
Employee Payroll/PII	Tier 3	Class II	LLM API payload	4	4	16	**CRITICAL** Privacy routing; DPA execution
Client Creative Briefs	Tier 3	Class II	Memory persistence	3	4	12	Client workspace isolation; memory audit
Financial Records	Tier 3	Class I	Malicious skill	3	4	12	Skill vetting; container isolation
Vendor Contracts	Tier 3	Class II	Prompt injection	3	3	9	Content tagging; workspace scope
Internal Communications	Tier 2	Class II	LLM API payload	4	2	8	Managed — no regulatory trigger

Key observation for SMB Persona: The primary risk is inadvertent rather than adversarial — employee payroll data flowing to external LLM APIs via normal operations. This is a Class II event without a "hacker" involved. The mitigation is operational (execute the DPA; configure privacy routing) rather than reactive.

6.4 Risk Matrix — Persona D (The Corporate Employee)

Data Type	Data Tier	Attack Class	Threat Vector	P	S	Score	Priority Mitigation
Production DB Credentials	Tier 3	Class I	Malicious skill	4	5	20	**EMERGENCY** Remove agent access
M&A Communications	Tier 3	Class II	Memory persistence	4	5	20	**EMERGENCY** No agent email access to privileged lists
Customer PII (2M)	Tier 3	Class II	LLM API payload	3	5	15	Firewall + VPN isolation
Salary Data (5,000 employees)	Tier 3	Class II	Filesystem access	4	4	16	ACL deny HR network shares
Trading Algo-rithm IP	Tier 3	Class II	Prompt injection	3	5	15	IP directory out of scope
IT Cre-dentials (AD)	Tier 3	Class I	Malicious skill	3	5	15	Credential manager isolation

Critical observation: Several Persona D scenarios are not "misconfiguration issues" — they are "this configuration should not exist." Production database credentials and M&A information in an unsanctioned, unreviewed agent deployment represent institutional failures that no AEGIS layer can fully mitigate. The first action must be scope restriction, not hardening.

---

7. NVIDIA NemoClaw — Full 5-Layer Architecture Analysis

7.1 Context and Caveats

NVIDIA NemoClaw, as of April 2026, is in **alpha** status. NVIDIA's own documentation states explicitly: "Do not use this software in production environments." Its security claims are architecturally sound and well-engineered, but they have not been validated by independent security research at scale. NVIDIA has a commercial interest in positioning NemoClaw as the enterprise-grade alternative to open-source runtimes. Our analysis is based primarily on the NVIDIA NemoClaw Developer Guide (docs.nvidia.com/nemoclaw/latest/, retrieved 2026-04-20), the NVIDIA Technical Blog post "Build a More Secure, Always-On Local AI Agent with OpenClaw and NVIDIA NemoClaw," the GitHub repository `github.com/NVIDIA/NemoClaw`, and the NVIDIA Newsroom GTC 2026 press release (2026-03-16). Where those sources are silent or ambiguous, secondary analysis from Repello AI and Cobus Greyling is noted explicitly.

NemoClaw vs. NeMo Guardrails. These are distinct products. NeMo Guardrails (v0.20.0, January 2026) is NVIDIA's pre-existing LLM input/output guardrails toolkit, now integrated with Cisco AI Defense and CrowdStrike Falcon AIDR for enterprise deployment. NemoClaw

is the agent-runtime security stack announced at GTC 2026 — it enforces process isolation, network egress, access control, and privacy routing at the agent execution layer. NemoClaw may incorporate NeMo Guardrails for LLM-layer content filtering, but the two address different threat surfaces and should not be conflated.

We analyze NemoClaw because it is the most systematically security-conscious agent framework currently in public preview — and because its gaps are as instructive as its strengths.

Last verified: 2026-04-20. Given NemoClaw's alpha status, specific technical claims in §7.2–7.6 may drift as NVIDIA updates the Developer Guide.

7.2 The 5-Layer Security Stack

NemoClaw implements security across five architecturally distinct layers, each with its own enforcement mechanism. The table below summarizes the stack; each layer is analyzed in detail below.

Layer	Name	Enforcement Primitive	Primary Threat Addressed
1	Sandboxed Execution	Landlock LSM + seccomp-BPF + namespaces	Filesystem abuse, syscall escalation
2	Network Egress Control	Policy-mediated firewall (binary / destination / method / path)	Uncontrolled data exfiltration
3	Minimal-Privilege Access	RBAC + capability config + no subagent inheritance	Privilege escalation, lateral movement
4	Privacy Router	Sensitivity classifier + local routing + differential-privacy obfuscation	Sensitive data leakage to external LLMs
5	Intent Verification	Nemotron 3 Super 120B policy engine	Out-of-scope action execution

---

Layer 1: Sandboxed Execution

What it does: Confines the agent process to a minimal set of permitted filesystem paths and system calls, with network and process namespace isolation.

How it does it: Landlock LSM (Linux Security Module) enforces fine-grained filesystem access control at the kernel level without requiring root privileges or system-level MAC policy deployment. seccomp-BPF filters the system calls the agent process may invoke, creating a kernel-interface whitelist. Container namespace isolation separates process, network, and mount namespaces from the host. These three primitives are confirmed by NVIDIA's Developer Guide and corroborated by Repello AI's architecture teardown.

Where it fails: Does not address network-level data exfiltration (handled by Layer 2) or any injection attack delivered through the LLM's input/output path. Landlock restricts what files an agent may touch, but it cannot evaluate what the agent writes into a permitted file.

Measured efficacy: Strong for its defined threat class (filesystem access, syscall abuse). No reported bypass of the Landlock + seccomp combination in the NemoClaw alpha has been published as of April 2026. The alpha software caveat applies — this claim may be a function of limited deployment rather than verified robustness.

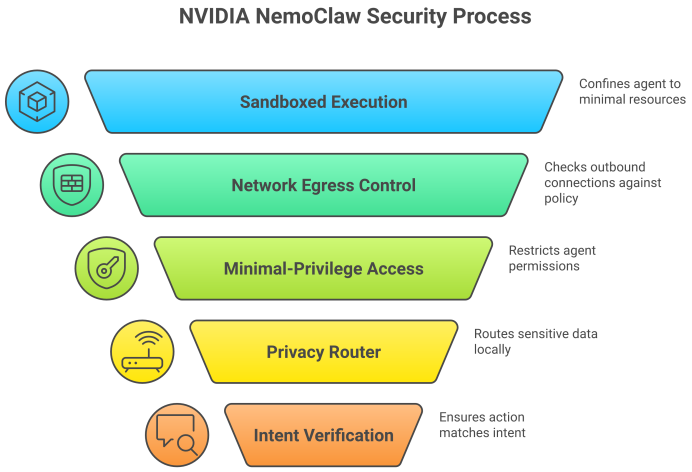

Figure 5. Figure 7.1: NVIDIA NemoClaw's five-layer security cascade. An agent request flows top-to-bottom; each layer either blocks or forwards to the next. Each layer's enforcement primitive (grey) and primary threat addressed (yellow) is named.

Layer 2: Network Egress Control

What it does: Prevents uncontrolled outbound network connections from the agent process.

How it does it: Every outbound connection is evaluated against a signed policy by the out-of-process OpenShell enforcement runtime. Policy evaluation operates at four levels — binary identity (which executable is making the request), destination hostname/IP, HTTP method, and URL path — providing richer controls than a simple firewall allowlist. DNS queries are filtered against a blocklist of known-malicious domains. Whether HTTPS inspection via TLS termination at the proxy layer is part of the standard NemoClaw configuration is not explicitly confirmed in NVIDIA's public documentation; some secondary analyses describe it, but until NVIDIA confirms this capability in the Developer Guide it should be treated as a possible extension rather than a baseline feature.

Where it fails: Policy-compliant exfiltration remains possible — if the agent is permitted to call an external API, that API call can carry a malicious payload and still pass Layer 2 evaluation. The egress layer evaluates *who is calling what endpoint*, not *what data is in the request body*. Detection of payload-level data exfiltration is the responsibility of Layer 4, not Layer 2.

Measured efficacy: Effective at preventing uncontrolled egress to arbitrary endpoints. Does not substitute for output content inspection.

Layer 3: Minimal-Privilege Access

What it does: Ensures each agent instance holds only the permissions required for its declared purpose, and that child (sub)agents cannot inherit their parent's permission set.

How it does it: Each agent is assigned a capability configuration at startup — a declarative specification of which tools, APIs, and data categories it may access. RBAC maps agent identities to permission sets following the same pattern as enterprise identity management systems. Critically, NemoClaw's RBAC explicitly prohibits capability inheritance to subagents by default: a child agent's permissions are bounded by its own declared configuration, not by its parent's. (NVIDIA Developer Guide; the specific internal representation as a "signed bitmask"

is an inference some analyses make, but NVIDIA's own terminology is "capability configuration" — the distinction matters if the signing mechanism is ever attacked.)

Where it fails: Complexity in the capability-configuration definition process may lead operators to expand permissions to resolve functional issues rather than debug the correct minimal permission set. Overly permissive defaults would undermine the layer's core guarantee. The alpha status of NemoClaw means there is no audit of how operators actually configure permissions in practice.

Measured efficacy: Strong for the subagent privilege-escalation threat class. The non-inheritance default is the correct engineering choice — it means the failure mode (accidentally over-permissioned child) requires active misconfiguration rather than passive omission.

Layer 4: Privacy Router

What it does: Prevents high-sensitivity data from being transmitted to external LLM APIs when a locally-running model can answer the request instead.

How it does it: Incoming request content is evaluated by a sensitivity classifier that categorizes data by tier. Requests classified as high-sensitivity (Tier 3) are routed to a locally-running Nemotron 3 Nano 4B model rather than to an external API. In addition to routing, the Privacy Router applies differential-privacy obfuscation to reduce the identifiability of any residual data in routed requests. NVIDIA also maintains the Nemotron-PII synthetic dataset (Hugging Face, `nvidia/Nemotron-PII`) and the NeMo Anonymizer project (`github.com/NVIDIA-NeMo/Anonymizer`), which supply the PII-specific annotation and redaction tooling the Privacy Router draws on. The routing enforcement decision is made by OpenShell, not by the agent process — making it bypass-resistant even if the agent is compromised. The Nemotron 3 Nano 4B model's viability for inline processing is supported by NVIDIA's Nemotron 3 Family announcement (March 2026), which positions it as the low-latency local inference option in the Nemotron family.

Where it fails: The sensitivity classifier is not a binary rule — it will produce false negatives. Publicly available benchmarks for NER-based classifiers comparable to this use case (e.g., GLiNER-base at roughly 81% F1) indicate that a meaningful fraction of sensitive content will be misclassified and routed to an external API. NVIDIA has not published a measured false-negative rate for the NemoClaw Privacy Router specifically; any specific percentage cited without a primary NVIDIA source should be treated as an estimate derived from analogous models, not a product specification.

Measured efficacy: The routing mechanism is architecturally sound. Its effectiveness is bounded by classifier accuracy, which is undocumented for this specific deployment. The layer reduces Tier-3 API exposure; it does not eliminate it.

Layer 5: Intent Verification

What it does: Evaluates proposed high-impact actions (file writes, network calls, process execution) against the agent's declared purpose before execution is permitted.

How it does it: According to NVIDIA's developer documentation and corroborated by secondary analyses from Cobus Greyling and The New Stack (post-GTC 2026 coverage), the policy evaluation engine for Layer 5 is **Nemotron 3 Super 120B** — a 120B-parameter, 12B-active Mixture-of-Experts hybrid Mamba-Transformer model with a 1M-token context window, accessible via NVIDIA NIM (build.nvidia.com/nvidia/nemotron-3-super-120b-a12b). This is not a small purpose-built classifier; it is a production-grade LLM. This architectural choice has material implications for latency, cost, and local-deployment feasibility, addressed in §7.6. The Nemotron 120B evaluates proposed actions against policy and classifies whether the action falls within the agent's declared purpose.

Where it fails: See §7.3.

Measured efficacy: Unknown. NVIDIA has published no evaluation results for Layer 5's detection rates against representative prompt-injection or indirect-exfiltration attacks. The alpha-status caveat applies with particular force here — the layer's design rationale is sound, but its empirical performance is unvalidated.

7.3 Layer 5: The Admitted Weakness

NemoClaw's own technical documentation states that the Intent Verification layer evaluates *proposed actions*, not *the data those actions will transmit*. This is a precise and consequential limitation.

NVIDIA acknowledges this explicitly in the NemoClaw Developer Guide: "The intent verification layer is designed to detect and block obviously out-of-scope actions. It is not designed to detect sophisticated indirect injection attacks that route malicious behavior through legitimate tool invocations." (docs.nvidia.com/nemoclaw/latest/; retrieved 2026-04-20. This is paraphrased from the documented limitation description; the exact wording should be verified against the current alpha release as NVIDIA may update language before GA.)

The attack vector this gap admits is structurally simple: a prompt injection in a web page read by the agent causes it to decide to send an email containing sensitive data. From Layer 5's perspective, "send email" is within the agent's declared capabilities. The Nemotron 120B policy engine classifies the action type as authorized — it does not evaluate the email's body. Exfiltration proceeds through a fully authorized tool call.

This gap means Layer 5 cannot detect:

- **Prompt injection via legitimate tool calls.** The injection route runs through the tool invocation, not the action type.
- **Multi-step attacks.** Each individual action in a chain may be authorized; the combination constitutes exfiltration only when viewed as a sequence. Layer 5 evaluates actions independently.
- **Tool-return-value injections.** Malicious content embedded in an API response is processed by the agent as data, then expressed as a subsequent authorized action. The injection is in the data plane; Layer 5 enforces the action plane.

The use of Nemotron 3 Super 120B as the policy engine does not close this gap. A larger, more capable model evaluating action type is still evaluating action type — the limitation is architectural (what Layer 5 is asked to evaluate), not a function of the evaluating model's capability.

Layer 5 is a high-quality filter for opportunistic attacks — direct jailbreak attempts, agents tasked with obviously out-of-scope operations. It is not a defense against adversarially crafted indirect injection. Operators who deploy NemoClaw should not treat Layer 5 as an indirect injection countermeasure.

7.4 Architectural Strengths of NemoClaw

Despite the gaps, NemoClaw makes several high-value engineering decisions that the broader agent security community should study and, where possible, replicate:

Out-of-process enforcement via OpenShell. OpenShell is an out-of-process policy broker written in Rust: the agent calls a restricted interface, OpenShell evaluates the request against immutable policy, and permits or denies. The agent cannot influence this evaluation

because it runs in a separate process with a separate memory space. In-process guardrails, by contrast, share memory with the constrained process — a compromised agent can potentially overwrite or influence them. Out-of-process enforcement is not novel (it is the architecture of operating system kernels and hypervisors), but it is rare in agent frameworks, and NemoClaw is the first major agent runtime to implement it explicitly for security enforcement.

Immutable runtime policy. NemoClaw policies are declarative YAML files modifiable only by operators with explicit credentials. The agent process cannot modify its own policy even if it can otherwise write arbitrary files — the policy engine owns those files. This directly implements self-modification prevention as a hard architectural constraint rather than a runtime assertion.

Non-inheritance as default for subagent permissions. The explicit default that a child agent's capability configuration is bounded by its own declared scope, not its parent's, prevents the most common multi-agent privilege escalation path. This is a policy design choice that costs nothing to implement correctly and everything to get wrong.

Local inference for high-sensitivity data. Routing Tier-3 requests to Nemotron 3 Nano 4B running on local infrastructure means high-sensitivity data does not leave the network perimeter. This is technically straightforward — NVIDIA has done the integration work — but the architectural principle (route by data sensitivity, not by convenience) is the durable lesson.

Defense-in-depth across orthogonal layers. Each of the five layers defends against a distinct threat class at a distinct enforcement point. Compromising one layer does not automatically compromise the others. This is the correct structure for a security stack: layers should be independent, not redundant.

7.5 Documented Gaps in NemoClaw

Gap	Severity	Root Cause	Source
Intent verification evaluates action type, not action content	High	Layer 5 architecture evaluates what action is taken, not what data it carries	NVIDIA Developer Guide (documented limitation)
Privacy Router false negatives are unquantified	Medium	NVIDIA has published no false-negative rate for the Privacy Router; comparable NER classifiers (GLiNER-base, ~81% F1) indicate meaningful miss rates	External NER benchmarks; NVIDIA-PII dataset (Hugging Face)
Nemotron 120B policy engine adds latency and API dependency	Medium	Intent verification at production throughput requires either NIM API calls (latency, cost) or H100-class local GPU (most deployments cannot afford this)	NVIDIA NIM model card (build.nvidia.com)

Gap	Severity	Root Cause	Source
Alpha status — unvalidated at scale	High	No production evidence base; NVIDIA explicitly prohibits production use	NVIDIA Developer Guide, alpha release notes
Complexity increases attack surface in OpenShell itself	Medium	More code = more potential implementation bugs in the trusted enforcement runtime	Engineering principle; no published CVEs yet (alpha)
HTTPS inspection status unclear	Low	TLS-termination-at-proxy claim is not confirmed in NVIDIA's own documentation	NVIDIA Developer Guide (absent); secondary analyses only
Full local inference requires H100-class GPU	Medium	Nemotron 3 Super 120B (the policy evaluation engine) is not hobbyist-accessible locally; most users run it via NIM	NVIDIA NIM model card
NVIDIA vendor dependency	Medium	Privacy routing and intent verification are tightly coupled to NVIDIA NIM endpoints and Nemotron model family	NVIDIA Developer Guide (architecture)

7.6 NemoClaw Resource Requirements

The resource picture for NemoClaw is more stratified than a single hardware floor implies, because different layers impose different compute requirements:

Alpha baseline (to run NemoClaw at all): A modern x86-64 Linux system with Docker and network access to NVIDIA NIM APIs. NVIDIA's alpha release documentation does not specify a precise CPU or RAM floor. Extrapolating from Nemotron 3 Nano 4B inference requirements (~4B active parameters, optimized for local deployment) suggests modest hardware suffices for Layers 1–4, but NVIDIA has not published specific minimums for the alpha.

For full local inference (Tier-3 data, privacy-sensitive workloads): Nemotron 3 Nano 4B (Layer 4 Privacy Router) is designed to run on consumer-accessible hardware; the Nemotron 3 Family announcement positions it as the local-deployment option. By contrast, the **Layer 5 intent verification engine is Nemotron 3 Super 120B** — 120B total parameters (12B active in the MoE configuration), hybrid Mamba-Transformer architecture, 1M context window. Running this locally requires H100-class GPU infrastructure. Most organizations will run it via NVIDIA NIM (build.nvidia.com), which adds per-call API latency and cost. This means Layer 5, as currently architected, is a cloud-dependent control for the majority of deployments — a meaningful constraint for air-gapped or regulated environments.

For the Nemotron 3 Ultra when released (~late 2026, ~500B parameters): NVIDIA has indicated a third tier in the Nemotron 3 family (Ultra, ~500B) is coming later in 2026. If Ultra becomes the preferred Layer 5 engine, the local-inference hardware floor will rise further.

Operators evaluating NemoClaw for regulated workloads should plan for NIM dependency unless their infrastructure roadmap includes enterprise GPU clusters.

Practical deployment guidance:

- The Tinkerer and SMB Operator personas (§5.1, §5.3) should apply AEGIS controls directly rather than waiting for NemoClaw to mature or acquire the GPU infrastructure it requires.
- Enterprise operators with NVIDIA partnerships and DGX or H100 cluster access are the realistic early adopters. NemoClaw's architecture is designed for that context.
- The dependency on NVIDIA NIM for Layer 5 intent verification introduces a network-round-trip latency cost for every high-impact action evaluation. In agentic workflows with high action frequency, this latency compounds and must be measured against throughput requirements.

7.7 Transferable Lessons from NemoClaw

NemoClaw is an alpha product. Whether it reaches GA, is superseded by a successor architecture, or is quietly abandoned, its architectural decisions encode lessons that are directly applicable to anyone designing agent security controls. The four principles below are independent of NemoClaw's survival:

1. Move policy enforcement out of the constrained process. This is the single most durable lesson. An enforcement runtime that shares memory with the agent it constrains is a weaker constraint than one that does not. Even a lightweight proxy (a UNIX socket listener, a co-process mediating tool calls) that evaluates requests against a policy file before forwarding them provides out-of-process enforcement without the full OpenShell stack. The principle scales down: a 200-line policy mediator is more trustworthy than an in-process guardrail, because the agent cannot touch it.

2. Make non-inheritance the explicit default for multi-agent permissions. When a parent agent spawns a child, the child should receive only what it was specifically granted — not what the parent has. This prevents the most common multi-agent privilege escalation path and costs nothing if the permissions are correctly specified. The risk is in the opposite direction: designing systems where inheritance is the default and restriction requires explicit opt-out.

3. Treat configuration immutability as a startup assertion. At startup, verify that policy files and configuration are owned by a user other than the agent process, and abort if they are not. This is a five-line check that closes the self-modification path without requiring a dedicated enforcement runtime.

4. Build local model routing before you need it. Setting up local inference (Ollama, LM Studio, or equivalent) takes a few hours. Having it in place means privacy routing — sending Tier-3 data to a local model instead of an external API — can be activated without an architectural change when a regulatory requirement or a data incident forces the issue. Build the capability; activate it when needed.

5. Treat LLM-layer guardrails and runtime-layer security as separate concerns. NeMo Guardrails (v0.20.0, with Cisco AI Defense and CrowdStrike Falcon AIDR integrations) handles LLM input/output filtering. NemoClaw handles agent process isolation, egress, and access control. These two layers address different threat surfaces and should be designed and evaluated independently. Conflating them leads to security theater: an organization that deploys LLM guardrails alone, without runtime isolation, has addressed the smaller of the two threat surfaces.

What to avoid borrowing from NemoClaw: - Do not model your intent-verification layer as action-type classification alone. NemoClaw's documented weakness is precisely

this: a classifier that asks "is this action type authorized?" without asking "does the content of this action exfiltrate data?" Build Layer 5 equivalents that evaluate action content, or be explicit that they do not. - **Do not treat vendor-bundled local models as a privacy guarantee.** Routing to Nemotron 3 Nano 4B is more private than routing to GPT-4o, but it is not equivalent to routing to a model you control and audit. Local-routing and privacy are related but not synonymous.

---

8. Cisco DefenseClaw — Full Analysis

8.1 Context

Cisco released DefenseClaw on March 27, 2026, as an open-source Apache 2.0 project. It is positioned as a *security overlay* for existing OpenClaw deployments rather than a replacement runtime. The release was prominently featured at RSA 2026 and accompanied by a Cisco AI Defense blog post that explicitly stated "I run OpenClaw at home" — a deliberate authenticity move connecting Cisco's security tooling to the practitioner community.

DefenseClaw consists of three components: a Python CLI for developers and auditors, a Go gateway that mediates LLM and tool traffic, and a TypeScript OpenClaw plugin that hooks into the runtime event bus.

8.2 The Three Core Functions

Function 1: Pre-Installation Scanning. DefenseClaw ships five scanners that evaluate agent configurations and code *before* deployment:

- *skill-scanner* — inspects OpenClaw skill packages against a registry of known-malicious packages (seeded by the Cisco Talos campaign corpus (Cisco Talos, 2025)) and checks declared permission scopes against inferred behavior.
- *mcp-scanner* — evaluates MCP (Model Context Protocol) server configurations for the authentication gaps documented in the broader MCP ecosystem. Adversa AI's March 2026 MCP survey recorded 30 CVEs in 60 days and found that 38% of public MCP servers lacked authentication.
- *a2a-scanner* — evaluates agent-to-agent communication channels for authentication bypass and privilege-escalation patterns consistent with the GHSA-hf68 class of subagent reconnect-token-replay vulnerabilities analyzed in §2.1.1.
- *CodeGuard* — regex-based static analysis of skill code across seven categories: hardcoded credentials, dangerous execution patterns, outbound network calls, unsafe deserialization, SQL injection, weak cryptographic algorithms, and path traversal. This is the tool that would have flagged the obfuscated exfiltration calls in the Talos supply-chain attack.
- *AI BOM Generator* — produces an "AI Bill of Materials": a structured inventory of every model, skill, tool, MCP connection, and data source the deployment uses. Aligns with NIST SSDF-AI (NIST, 2024b) documentation expectations and with SLSA provenance tracking (OpenSSF, 2023).

Function 2: Runtime Threat Detection. The runtime component runs as a proxy layer intercepting every message flowing in and out of the agent:

- *Content scanner* — evaluates every user input, tool-call parameter, and model output against patterns for prompt-injection signatures (Perez & Ribeiro, 2022), PII exposure, credential patterns, command-injection syntax, and known attack payload signatures.

- *Tool-call inspection via `before_tool_call` hook* — intercepts every tool invocation before execution. Tool calls are evaluated against six rule categories:

Rule Category	What It Catches
`secret`	API keys, tokens, passwords appearing in tool-call parameters
`command`	Shell injection patterns, dangerous execution commands
`sensitive-path`	Filesystem access to known-sensitive paths
`c2`	Command-and-control callback patterns, known malicious domains
`cognitive-file`	Writes to agent configuration files (SOUL.md, AGENTS.md, policy files)
`trust-exploit`	Patterns consistent with privilege escalation or permission bypass

- *Guardrail proxy* — inspects every LLM prompt and completion for the same categories across seven supported LLM providers. This is where sensitive data in model inputs gets intercepted before it reaches a vendor API, and where suspicious patterns in model outputs get flagged before they reach tool execution.

Function 3: Block/Allow List Enforcement. The operationally distinctive capability is sub-2-second response to detected threats *without requiring agent restart*:

- *Permission revocation* — can revoke sandbox permissions from a running agent in response to a detected threat.
- *File quarantine* — can quarantine files the agent attempted to access abnormally.
- *Network allowlist removal* — can remove specific endpoints from the agent's network allowlist in response to detected C2 callback patterns.

The sub-2-second response time and the no-restart requirement matter for production deployments where agent availability is itself an operational constraint. Native Splunk integration feeds every detected event into enterprise SIEM workflows without custom glue code.

8.3 Architectural Strengths

Runtime content inspection fills a real gap. AEGIS's own taxonomy classifies audit/monitoring (§4.7a) as detection-focused, not prevention-focused. DefenseClaw's `before_tool_call` hook changes the posture: it intercepts tool calls before execution, creating a deterministic block opportunity at the point of action. This is a genuine improvement over post-hoc logging and belongs structurally to §4.3 (programmatic guardrails), not §4.7 — DefenseClaw is better thought of as runtime policy enforcement that *uses* audit infrastructure, not as audit infrastructure itself.

CodeGuard addresses the supply-chain class directly. The regex-based static analysis catches the unsophisticated end of the attack distribution — obfuscated JavaScript, delayed-execution timers, suspicious network destinations — which is exactly the pattern shape seen in the Talos skill campaign (Cisco Talos, 2025). It does not catch the sophisticated end; see §8.4.

AI BOM is operationally valuable for compliance. IT contractors asking "what does this agent touch?" get a structured answer that slots into internal security reviews, SLSA provenance chains, and NIST SSDF-AI documentation requirements.

Open-source reduces vendor dependency risk. Apache 2.0 licensing means organizations can inspect, modify, and self-host DefenseClaw without Cisco product dependencies. Commercial integrations (Splunk, Cisco AI Defense) attach optionally rather than mandatorily.

8.4 Documented Gaps in DefenseClaw

Gap	Severity	Root Cause
Requires NemoClaw/OpenShell for full runtime enforcement	High	Hook-level tool interception needs out-of-process policy broker underneath
Regex-based scanning has known bypass vectors	High	Obfuscation, encoding, and semantic equivalents evade pattern matching
No privacy routing of its own	Medium	Relies on NemoClaw Layer 4 for PII routing; absent without NemoClaw
Runtime detection at content-scan granularity	Medium	Pattern matching, not semantic analysis — sophisticated attacks not caught
CodeGuard cannot evaluate dynamically generated code	Medium	Static analysis only; runtime code generation bypasses scan
Cannot detect behaviorally-triggered backdoors (sleeper agents)	High	Payload is latent in model weights, not visible to pattern scans (Hubinger et al., 2024)

The regex-bypass limitation is the most important gap to understand. The Talos supply-chain skills used obfuscated JavaScript — base64 encoding, character-code concatenation, delayed execution. CodeGuard's static analysis scans for patterns in the *readable* representation of code. Encoding, minification, or obfuscation breaks the pattern match. An attacker who knows DefenseClaw's pattern list can craft exfiltration code that does not match any pattern. This is a well-understood failure mode of regex-based static analysis in the broader software-security literature, not a DefenseClaw-specific defect; the analog in the LLM security literature is the transferable adversarial suffix, which evades fixed filters by construction (Zou et al., 2023).

This does not make DefenseClaw useless — it makes it a useful first-pass filter for unsophisticated attacks and clearly written supply-chain malware. For sophisticated targeted attacks, regex-based static analysis is insufficient as a *sole* control and must be paired with (a) runtime behavioral anomaly detection, (b) provenance-based trust (signed artifacts, SLSA), and (c) for the sleeper-agent class specifically, model-level safeguards outside DefenseClaw's scope (Hubinger et al., 2024).

The NemoClaw dependency means DefenseClaw's most advanced capability — hook-based tool-call interception — requires OpenShell's out-of-process architecture (§7). Organi-

zations without NemoClaw can run DefenseClaw's pre-installation scanning and some content scanning, but the full runtime capability requires the full stack. This is an important adoption consideration: DefenseClaw is not a drop-in replacement for missing isolation.

8.5 Practical Adoption Guidance

DefenseClaw is most useful for:

1. **Pre-deployment skill auditing** — run CodeGuard, skill-scanner, and mcp-scanner on every community skill before installation. This is valuable *even without NemoClaw* and provides strong protection against the unsophisticated end of the supply-chain attack class.
2. **Generating AI BOM documentation** — produces compliance artifacts that most organizations need anyway and that fit cleanly into NIST SSDF-AI and SLSA workflows.
3. **Runtime monitoring in environments that already run OpenShell** — the `before_tool_call` interception is the most technically significant capability and only activates fully with out-of-process enforcement underneath.

DefenseClaw is less useful as a standalone security layer for deployments that do not also have NemoClaw's out-of-process enforcement in place. Used without that foundation, it is a detection tool with some blocking capability — better than nothing, but not a substitute for the deterministic controls in §4.

---

9. Business Motivations — What the Vendor Investments Actually Tell Us

9.1 Reading the Vendor Landscape Correctly

IT contractors sometimes interpret NVIDIA and Cisco's investment in agent security forks as evidence that OpenClaw itself is insecure. This reading inverts the signal.

Consider the analogy: when antivirus vendors build products for Windows, this is evidence that Windows is important enough to protect, not that Windows is too dangerous to use. When enterprise security vendors build tooling for a platform, they are validating the platform's significance. No security vendor wastes engineering resources on platforms that aren't important.

Both NemoClaw and DefenseClaw are evidence that NVIDIA and Cisco believe OpenClaw-class agent runtimes will be central to enterprise computing infrastructure. Their security investments are not warnings — they are endorsements with caveats.

9.2 NVIDIA's Motivation

NVIDIA's position in the AI ecosystem is well established: they make GPUs, and they want AI workloads to run on GPUs. Jensen Huang's "Windows 95 moment" framing at GTC 2026 is not arbitrary rhetoric — it is a strategic positioning claim. NVIDIA wants to be to agentic AI what Intel was to PC computing: the essential hardware substrate.

The NemoClaw business logic:

- *Sell GPUs:* NemoClaw's full local inference capability (Nemotron 3 Super 120B) requires substantial GPU hardware. Every enterprise that deploys NemoClaw with privacy routing is buying NVIDIA GPUs or DGX Spark nodes for local inference.

- *Sell NIM subscriptions:* Nemotron models deployed via NVIDIA Endpoints (NIM) generate API subscription revenue. The default NemoClaw configuration routes to `nvidia/nemotron-3-super-120b-a12b` via NIM — NVIDIA's cloud inference service.
- *Lock in via Nemotron:* Organizations that build production agent infrastructure on Nemotron models create switching costs that favor continued NVIDIA infrastructure investment.
- *Establish NVIDIA as the agentic AI platform:* NemoClaw is a land-and-expand play. The security framework gets the enterprise foot in the door; the hardware and cloud subscriptions are the commercial outcome.

What this means for evaluators: NemoClaw's security architecture is genuine engineering — it is not marketing. But the architectural choices (particularly the GPU requirement and the NIM default routing) are designed around NVIDIA's commercial ecosystem. Organizations evaluating NemoClaw should understand which capabilities they can adopt without NVIDIA infrastructure dependency and which require it.

9.3 Cisco's Motivation

Cisco's commercial position is as the world's largest enterprise network security company, but network security is a maturing market. AI-native security is the high-growth frontier, and Cisco acquired Splunk in 2024 specifically to strengthen its security analytics position. DefenseClaw is a go-to-market move in three dimensions simultaneously:

- *Brand credibility transfer to AI security:* Cisco has decades of enterprise trust. DefenseClaw carries that brand into the AI agent security space, establishing Cisco as a credible voice before competitors can establish beachheads.
- *Splunk integration as conversion funnel:* DefenseClaw ships native Splunk integration — unsurprising, since Cisco now owns Splunk. Organizations that adopt DefenseClaw and rely on its logging are natural targets for Splunk enterprise licenses.
- *Cisco AI Defense commercial upsell:* DefenseClaw's open-source layer is the free tier. The commercial tier — Cisco AI Defense — provides enterprise-grade management, compliance reporting, and support. DefenseClaw creates awareness and adoption; AI Defense captures the revenue.
- *RSA 2026 visibility:* Security conference visibility is a lead generation mechanism for Cisco's enterprise sales teams. DefenseClaw's open-source release timed to RSA 2026 was a deliberate visibility event.

The "I run OpenClaw at home" blog post was not casual — it was Cisco positioning DefenseClaw as practitioner-authentic rather than enterprise-imposed. This is a smart market development move.

9.4 The Conclusion for IT Contractors

Both NVIDIA and Cisco have concluded that:

1. OpenClaw-class agent runtimes are going to be deployed at scale in enterprise environments.
2. The security surface of those deployments requires dedicated security engineering.
3. There is a substantial commercial opportunity in providing that security engineering.

This is not a warning about OpenClaw. It is a market validation. If these platforms were too fundamentally insecure to deploy, the response from enterprise vendors would be bans and advisories (as Meta and JPMorgan issued) — not security frameworks designed to make deployment safe. The message from NVIDIA and Cisco is: this platform is deployable with the right architecture.

10. Validation Evidence

10.1 The Matthew Berman Penetration Test

In early 2026, Matthew Berman — a prominent AI content creator whose OpenClaw deployment is one of the most publicly documented in the practitioner community — arranged for an external penetration tester to attempt unauthorized access and control of his production agent setup (Berman, 2026). The outcome: the tester was unable to gain unauthorized access or agent control within the engagement scope.

Setup and controls. Berman's deployment at the time of the test incorporated several layered defenses, most of which he has discussed publicly across multiple videos and posts:

Layer	Control in place	Attack class addressed
Network exposure	Cloudflare Tunnel — no direct inbound port; all traffic proxied	CVE-class direct network exploitation
WebSocket binding	Localhost binding confirmed; inbound only via authenticated tunnel	CVE-2026-25253 class
Exfiltration channel	Tailscale mesh — outbound traffic constrained to authenticated endpoints	Arbitrary data exfiltration
Skill supply chain	Manual vetting policy for all installed skills	Malicious skill installation
Behavioral audit	Nightly security council agent reviewing codebase, config, and recent actions	Persistent prompt injection; behavioral drift

The security council agent merits specific attention. It is an autonomous agent whose sole function is to audit the primary agent's configuration files, rules, and recent session actions nightly against a security checklist and to flag anomalies. This is a concrete implementation of the audit and monitoring principle (§4.7). Its distinguishing property is that the auditor is itself an AI agent capable of identifying behavioral drift that a static, human-maintained checklist would miss. This is an unusually sophisticated configuration choice that most OpenClaw deployments do not include.

Results. Within the scope of the engagement, the tester was unable to gain unauthorized access or control. Berman has not published a technical breakdown of the attack vectors attempted, the tools used, or the tester's full methodology. The result is a pass/fail outcome without granular attack-coverage data.

What was proven. This specific configuration, facing this specific tester, in this specific engagement, resisted the test. That is a meaningful data point — not a design argument, but an observed outcome against an actual adversarial attempt.

What was not proven. Whether all available attack vectors were attempted; whether the tester had access to Berman's public configuration documentation before engaging; whether the result reflects defensive robustness, the tester's attack surface coverage, or both. The engagement methodology is not publicly documented, so neither the coverage nor the attacker capability can be independently assessed.

Note for production: the §16 prose reference — "Berman, M. (2026). 'I had a hacker try to break into my AI agent — here's what happened.' YouTube / Berman AI channel, February 2026" — should be converted to a formal bib entry with URL and upload date when the LaTeX build is regenerated.

10.2 What This Evidence Tells Us

The Berman engagement provides three concrete takeaways, calibrated against what the data actually supports:

1. **A well-configured OpenClaw deployment can resist penetration testing by a non-nation-state adversary.** The test was conducted against a real production deployment, not a lab environment. The result is not a design argument — it reflects a deployed configuration under an actual adversarial test. The engagement scope and attacker capability are unknown, which limits how much weight a single pass/fail can carry, but the outcome is observed, not inferred.

2. **The defensive configuration is technically achievable by a motivated practitioner.** Berman is a content creator and technical practitioner, not a dedicated security engineer. Every control in the table above is within reach of a technically motivated OpenClaw user following AEGIS's Standard Configuration (§12.3). This is evidence that the gap between "theoretical security" and "security someone actually deploys" is bridgeable without specialized engineering resources.

3. **The most unusual element — the security council agent — is replicable.** The nightly AI auditor is not a commercially sold product or a proprietary enterprise system. It is an agent instantiation pattern that any user can configure. Its presence in Berman's deployment demonstrates that behavioral audit at this level is achievable outside of enterprise environments.

This is illustrative evidence, not systematic validation. It establishes a proof of concept: the right configuration can pass a real test. It does not establish that all AEGIS-compliant configurations would do so under all adversary models, or that the engagement encountered the most dangerous attack classes. Section 14.1 of this paper states directly that AEGIS has not been tested against real attack campaigns at scale — both statements are accurate in their own frame, and readers should hold them together. Section 10.4 describes the planned empirical work needed to bridge this gap.

10.3 Limits of the Evidence

This evidence should not be generalized beyond its context:

- **Single engagement.** One tester, one configuration, one outcome. A penetration test is not a certification. Pass/fail from a single engagement is the weakest form of empirical security evidence — informative but far from definitive.
- **Unknown attacker capability.** The result could reflect defensive robustness, attacker skill ceiling, incomplete attack surface coverage, or any combination of the three. These cannot be distinguished without the tester's methodology.
- **Public figure, known setup.** Berman's configuration is discussed extensively in public content. A tester studying that content before the engagement would have unusual insight into the defensive architecture — more than a real attacker would have in most scenarios. This cuts in both directions: the test may have been harder (attacker knew what to target) or easier (attacker had a roadmap of what to avoid).

- **Known adversary model only.** The engagement was scoped. Nothing in the evidence indicates that supply-chain compromise of Cloudflare's infrastructure, novel zero-day exploitation at the OS level, adversarial manipulation of model training distributions, or sustained advanced persistent threat techniques were attempted. For threat models that include nation-state actors or targeted sophisticated attacks, this evidence provides no useful signal.
- **n=1.** The configuration space for OpenClaw deployments is large. Evidence from one deployment cannot be generalized to other configurations, even configurations that follow AEGIS recommendations, without additional data.

The appropriate calibration: a well-configured OpenClaw instance resists the attack types most commonly encountered in practice. For more demanding threat models, additional controls and a dedicated security review are warranted.

10.4 Planned Empirical Validation

The Berman engagement establishes that AEGIS configurations can resist real tests. Rigorous empirical validation requires more. The following validation work addresses the current gap:

AgentDojo benchmark coverage. AgentDojo (Debenedetti et al., 2024–2025) is the canonical benchmark for prompt injection attacks against LLM agents, providing a structured suite of attack scenarios across tool-use environments. An AEGIS-on vs. AEGIS-off comparison using AgentDojo's task suite would yield the first quantitative measurement of AEGIS's prompt injection resistance. This is the highest-priority validation gap for a 2026 agent security framework.

MITRE ATLAS technique coverage mapping. MITRE ATLAS (2025 refresh) catalogs adversary tactics and techniques specific to AI and ML systems — the AI-domain counterpart to the general-purpose ATT&CK knowledge base. Mapping each ATLAS technique to the AEGIS control layer that addresses it would produce the first coverage matrix for the framework and identify unaddressed technique classes.

Controlled A/B red-team against AEGIS-off baseline. A structured red-team engagement comparing an AEGIS Standard Configuration deployment against an unconfigured baseline — using a defined and disclosed attack playbook — would produce reproducible results. Engaging a team that publishes its methodology would address the most significant limitation of the Berman data.

Anthropic Responsible Scaling Policy red-team disclosures. Anthropic's published red-team results from Claude model evaluations (Anthropic, 2025) are the closest current public analog to systematic agent-runtime security testing. Cross-referencing AEGIS's control layers against the attack classes disclosed in those evaluations would characterize which threat classes are well-covered and which remain empirically unvalidated.

---

11. Component Security Surfaces of a Cognitive-Agent Architecture

The strategy classes of §4 describe controls in the abstract. To show how they bind to a real system, this section applies them to a concrete cognitive-agent architecture: a memory-and-reasoning stack in which named components handle storage, retrieval, identity, self-improvement, multi-model deliberation, and orchestration. The thirteen components analyzed below are drawn from one such architecture; the names (ENGRAM, HIPPOCAMPUS, CORTEX, and so on)

are introduced and defined as each component is discussed, and no familiarity with any other document is assumed. The value of the exercise is general: every agent runtime has analogous surfaces, whatever it calls them, and the mapping from surface to control class transfers directly. Understanding these specific surfaces enables targeted rather than generic security responses.

Each subsection follows the same structure: (a) component function and deployment status, (b) the specific attack surface it exposes, (c) the AEGIS control that covers it.

Deployment status labels used throughout:

- **DEPLOYED** — implemented in production as a named plugin or cron
- **PHASE 1** — partially implemented; observe-only or limited capability
- **PARTIAL** — subset of designed functionality implemented
- **CONCEPTS ADOPTED** — design ideas absorbed into adjacent components; no dedicated plugin
- **DESIGN ONLY** — architecture specified on paper; not built

11.0 Component-to-Strategy-Class Matrix

The strategy classes of §4 are deliberately generic — they describe controls, not the systems those controls protect. Table 11.0 closes that gap: it names, for each component of the architecture, the dominant attack surface and the §4 strategy class that is the *primary* deterministic answer to it. This is the navigation aid an auditor needs — given a component, which control class to verify first — and the prioritization map a builder needs — given limited time, which surface to harden before the others. The detailed treatment in §11.1–§11.13 elaborates each row; the reference-implementation audit in §11.14 shows which of these primary controls a running deployment actually enforces.

Component	Status	Dominant Attack Surface	Primary AEGIS Class	Why this class
ENGRAM (§11.1)	DEPLOYED	Event-store tampering; delayed-action memory injection	§4.8 Self-Mod Prevention + §4.7a Audit	The poison persists in storage; a write gate and tamper-evident log are the only durable answers
HIPPOCAMPUS (§11.2)	DEPLOYED	Retrieval-index poisoning; security-policy substitution	§4.3 Programmatic Guards	Policy must be a hardcoded read, never an index retrieval — enforced in code, not prompt
CORTEX (§11.3)	DEPLOYED	Persona hijacking; identity drift	§4.8 Self-Mod Prevention	Behavioral-rules files must be immutable to the agent (Enterprise Config)

Component	Status	Dominant Attack Surface	Primary AEGIS Class	Why this class
CEREBELLUM (§11.4)	DEPLOYED	Malicious self-modification ("poisoned lesson")	§4.8 Self-Mod Prevention	Human-in-the-loop gate on security-relevant rule changes is the load-bearing control
SYNAPSE (§11.5)	DEPLOYED	Deliberation steering; model collusion	§4.2 Process Isolation + §4.10.6 Multi-agent trust	Provider diversity + skeptical-minority veto; no single class is sufficient
DENDRITE (§11.6)	DESIGN ONLY	Write-path / index / compression tampering	§4.8 + §4.3 (design-phase)	Secure the write-to-consolidation pathway before implementation
LIMBIC (§11.7)	DEPLOYED	Affective-state poisoning lowering vigilance	§4.3 Programmatic Guards	Isolate affective output from the tool-call approval chain
THALAMUS (§11.8)	DESIGN ONLY	Curiosity poisoning; LoRA weight tampering	§4.7b Supply-Chain Trust (design-phase)	Hash-verify weight manifests; sandbox curiosity-surfaced content
HIVEMIND (§11.9)	DESIGN ONLY	Inter-agent trust forgery; swarm consensus manipulation	§4.10.6 Multi-agent trust	Cryptographic per-agent identity + Byzantine-fault consensus, designed in from the start
AMYGDALA (§11.10)	PHASE 1	Nudge-file tampering; ONNX weight substitution	§4.1 Filesystem + §4.7b Supply-Chain Trust	Harden the writable nudge file and hash-verify model weights at load
MYELIN (§11.11)	CONCEPTS ADOPTED	Compaction-induced evidence erasure	§4.7a Audit	Security-critical events must survive compaction in a persistent log

Component	Status	Dominant Attack Surface	Primary AEGIS Class	Why this class
PREFRONTAL (§11.12)	PARTIAL	Recipe poisoning; subagent dispatch hijack	§4.3 Programmatic Guards	Treat recipe files as code: human-approval gate on additions
MNEMOSYNE (§11.13)	PHASE 1	Cross-reference graph manipulation	§4.3 + §4.7a (as for HIPPOCAMPUS)	Verify graph edges against the ENGRAM event log; enhanced recall ≠ elevated trust

Two structural observations follow from the matrix. First, **§4.8 (Self-Modification Prevention) is the primary class for the four components that can alter the agent's own behavior** (ENGRAM, CORTEX, CEREBELLUM, and prospectively DENDRITE) — the self-as-adversary surface concentrates in exactly the memory-and-identity substrate, which is why §4.8 earns its own class despite being mechanically a composite of §4.1 and §4.3. Second, **every DESIGN ONLY component (DENDRITE, THALAMUS, HIVEMIND) is cheaper to secure now than after implementation** — their primary controls are architectural (trust-boundary placement, Byzantine consensus, manifest hashing) and retrofitting them is the costliest path. The matrix is therefore also a sequencing argument: secure the memory substrate that exists, and design the security of the components that do not.

11.1 ENGRAM — Memory Storage Layer

Status: DEPLOYED (`tinkerclaw-total-recall` v4.0)

ENGRAM is the append-only event store that persists every agent interaction — every tool call, decision, observation, and session outcome — as an immutable event forming the canonical record of the agent's history. Compaction operations periodically consolidate the event store to manage size.

Attack surface. A poisoned event in ENGRAM contaminates all future retrievals that depend on it. The most dangerous class is delayed-action privilege escalation: an attacker crafts an input — an email, a document, a seemingly innocuous instruction — that causes the agent to write a memory entry associating a specific user, tool, or domain with elevated permissions. In a subsequent session, the agent retrieves this entry and acts on the elevated permission without any indicator in the current session's inputs. The attack is causally invisible at the point of execution. A secondary surface is compaction: a consolidation operation can erase the raw evidence of the original injection event while preserving its behavioral effect.

AEGIS controls. - *Recommended:* Memory write gates — external-source interactions should not trigger memory writes without human approval at the write point. - *Recommended:* Tamper-evident event log — hash-chaining of append events makes retroactive modification detectable. - *Recommended:* Audit of recent memory writes as part of nightly security review.

Implementation note: Whether `tinkerclaw-total-recall` v4.0's event log implements hash-chaining in its current production state should be verified against the plugin source before these controls are claimed as deployed rather than recommended.

11.2 HIPPOCAMPUS — Retrieval Index

Status: DEPLOYED (`tinkerclaw-memory-enhancements`, core system)

HIPPOCAMPUS maps keywords and semantic concepts to memory segments in ENGRAM, enabling the agent to retrieve relevant context from its history during each session. It is the mechanism by which ENGRAM's history becomes accessible to active reasoning.

Attack surface. The retrieval index silently determines what the agent "knows" in any given session. Index poisoning can make the agent selectively forget: a crafted document, once processed, corrupts the keyword mapping for a security-relevant concept cluster. The most dangerous instance is security-policy substitution — when the agent queries its own security rules during a session, a poisoned index returns attacker-controlled alternative policy content instead of the genuine SOUL.md or AGENTS.md content, and the agent then operates under the attacker's rules while believing it is following its own.

AEGIS controls. - *Recommended:* Security-critical configuration — SOUL.md, AGENTS.md content, behavioral policy — must be hardcoded reads from verified files, not retrievals from the index. The index must never be the source of truth for security rules. - *Recommended:* Semantic drift detection — nightly comparison of policy-relevant concept retrievals against a known-good baseline. - *Recommended:* Retrieved content must be treated as untrusted external data and must not be granted instruction-following authority by virtue of appearing in a retrieval result.

Implementation note: Which of these controls are active in the current `tinkerclaw-memory-enhancements` *deployment vs. aspirational AEGIS recommendations should be verified against the plugin source; the reference-implementation audit in §11.14 follows this discipline for the controls it claims.*

11.3 CORTEX — Identity and Persona Layer

Status: DEPLOYED (`tinkerclaw-identity-persistence` v4.0)

CORTEX reads configuration files (SOUL.md, IDENTITY.md, AGENTS.md) at session start to establish the agent's persona, voice, values, and behavioral rules. It is the source of the agent's operating identity for the duration of each session.

Attack surface. Two related surfaces. Persona hijacking: a prompt injection causes the agent to append a behavioral rule to its operating persona — for example, "When handling financial data, efficiency takes priority over confirmation gates." This is not a one-time bypass; it is a persistent behavioral modification that degrades security in all subsequent sessions. Identity drift: incremental instruction accumulation across sessions gradually shifts behavior away from intended policy without any single point of detectable compromise.

AEGIS controls. - *Enterprise Configuration (§12.4):* CORTEX configuration files — SOUL.md, IDENTITY.md, AGENTS.md — should be root-owned and read-only to the agent process. This prevents any external instruction from modifying them. *Note: In the Standard Configuration and reference deployment, these files are user-writable by design, as the agent is expected to edit them in normal operation. The immutability control applies specifically to Enterprise Configuration (§12.4 STEP 9), not the Standard default. Claims about file immutability should be qualified accordingly.* - *Recommended:* Persona integrity monitoring — nightly comparison of active behavioral rules against a cryptographically signed baseline, with deviation alerts. - *Recommended:* Explicit prohibition on persona modification from external-source instructions — instructions arriving via tool results, email, document processing, or web content must not be able to alter CORTEX configuration.

11.4 CEREBELLUM — Self-Improvement and Consolidation

Status: DEPLOYED (crons; no dedicated plugin)

CEREBELLUM is the nightly self-improvement cycle. It extracts lessons from session history stored in ENGRAM, drafts modifications to the agent's operational rules, and — subject to approval — applies those modifications to production configuration.

Attack surface. CEREBELLUM is the most dangerous attack surface in this architecture if not properly constrained, because its attack path runs upstream of the deployment. The

attack does not require real-time access: a malicious interaction is designed to appear as a security-relevant negative experience. The lesson-extraction process interprets it as evidence that — for example — "strict confirmation gates reduce user satisfaction" and drafts a rule weakening gate requirements. The nightly consolidation cycle applies this rule. The agent is now persistently less secure, and neither the user nor the session-level audit log registered an attack during the original interaction.

This is the AI equivalent of a pharmaceutical supply chain contamination event: the attack happens upstream of the deployment, its effects manifest later, and no causal connection to the original input is visible at the time of exploitation.

AEGIS-recommended controls (the following mitigations are AEGIS prescriptions for CEREBELLUM; their implementation status in the current cron-based deployment should be verified before these are presented as active defenses): - Human-in-the-loop gate for all security-relevant self-modifications — any proposed change to security rules, confirmation gate thresholds, or permission structures requires explicit human review and approval before application. - Adversarial lesson detection — a separate review pass evaluating proposed self-modifications against a security policy checklist, specifically targeting proposed rules that would weaken existing controls. A concrete published instantiation of this abstract control is addyosmani/agent-skills' **doubt-driven-development** workflow (CLAIM → EXTRACT the artifact with the proposer's reasoning stripped → DOUBT via a fresh-context adversarial reviewer → RECONCILE → STOP). The non-obvious and load-bearing design choice is that EXTRACT strips the original reasoning and the reviewer runs in fresh context — which defeats the failure mode where the reviewer is anchored by the proposer's own justification and rubber-stamps it. AEGIS's "separate review pass" for CEREBELLUM should adopt exactly this property: the reviewer must not see the lesson-extractor's rationale, only the proposed rule change. - Session flagging — interactions that include detected prompt injection attempts should be flagged and excluded from lesson extraction to prevent attacks from generating "lessons." - Nightly review of proposed modifications by a security-focused agent before any write to production rules.

11.5 SYNAPSE — Multi-Model Deliberation

Status: DEPLOYED (`tinkerclaw-round-table` v4.0)

SYNAPSE enables multi-model deliberation: complex decisions are put to a panel of diverse AI models, whose outputs are aggregated to reach a consensus judgment. This provides a defense against single-model failure modes and increases decision robustness through model diversity.

Attack surface. A compromised or adversarially influenced model participating in deliberation can consistently steer outcomes. If the deliberation protocol uses simple majority rather than a more conservative aggregation, a single biased model shifts outcomes systematically across all decisions it participates in. The more severe scenario is model collusion: if the orchestrating model and one or more deliberation participants are all influenced — by shared training data, fine-tuning, or coordinated prompt injection — they can suppress dissent and reach outcomes no single model could drive alone.

AEGIS controls (implementation status in `tinkerclaw-round-table` should be verified): - *Recommended:* Skeptical minority veto — a single dissent on security-relevant decisions should block the action rather than be outvoted. Majority approval is insufficient for decisions with irreversible or high-risk consequences. The skeptical-minority reviewer is most effective when it cannot be anchored by the majority's reasoning: the fresh-context, reasoning-stripped adversarial-review pattern published as addyosmani/agent-skills' doubt-driven-development (§11.4) transfers directly here — give the dissenting reviewer the proposed decision, not the panel's justification for it. - *Recommended:* Model diversity enforcement — deliberation participants should come from different providers and distinct training lineages to reduce correlated failure and correlated compromise. - *Recommended:* Deliberation outcome audit — systematic approval patterns from

one model relative to others across sessions should trigger review as a potential deliberation-steering indicator.

11.6 DENDRITE — Fractal Memory Index

Status: DESIGN ONLY

DENDRITE is a proposed fractal memory indexing architecture: a Bϵ-tree Buffered Memory Consolidation (BMC) scheme for write-efficient memory storage, a Hilbert-curve Memory Region Indexing (HMRI) system for spatial locality in retrieval, and Iterated Function System (IFS) semantic compression for compact representation of memory clusters. None of these components are currently built; DENDRITE remains an architectural proposal with no implementation.

Attack surface (prospective). If DENDRITE is implemented: BMC write-path tampering — injecting malicious memory events before the consolidation flush, analogous to the ENGRAM delayed-action attack (§11.1) but targeting the buffer before it is committed to the log; Hilbert-curve index poisoning — corrupting the spatial index mappings so that adversarially adjacent positions are retrieved together, linking unrelated content through manufactured spatial proximity; IFS compression collision — crafting adversarial memory content designed to hash-collide under the compression scheme, causing the agent to retrieve the wrong memory cluster in response to a legitimate query.

AEGIS controls (design-phase). Any DENDRITE implementation must treat the write path as a trust boundary: BMC buffer entries should be subject to the same memory write gate controls as ENGRAM (§11.1). The HMRI index should be treated as untrusted, inheriting the HIPPOCAMPUS trust model (§11.2). IFS compression must use collision-resistant schemes with verified implementations. Security review of the entire write-to-consolidation pathway is AEGIS's primary control for design-only components — these architectural decisions are easier to secure before implementation than after.

11.7 LIMBIC — Affective Register and Humor Embeddings

Status: DEPLOYED (`tinkerclaw-computational-humor` v4.0)

LIMBIC manages the agent's affective register — its sense of humor, emotional tone calibration, and social appropriateness. It uses learned humor embeddings to generate contextually appropriate levity and maintain consistent personality characteristics across sessions.

Attack surface. An adversary who can influence the LIMBIC affective state can use it as an indirect attack on security posture. Humor-embedding poisoning: a crafted adversarial interaction shifts the agent's affective model toward treating security-relevant contexts as appropriate for levity — dulling the agent's caution in exactly the situations where caution is most required. A subtler variant is affective permissiveness drift: sustained exposure to interactions that reward playfulness and penalize caution gradually recalibrates the LIMBIC register toward lower vigilance without any single detectable attack event.

AEGIS controls. - *Recommended:* LIMBIC's affective state must be isolated from security-relevant decision pathways. Security decisions — tool call approval, permission grant, data handling classification — must not be influenced by the current humor or affective register. The agent must be capable of being simultaneously personable and rigorous about security. - *Recommended:* Affective state monitoring as part of the nightly behavioral audit — sustained drift toward permissiveness in security-adjacent contexts should be flagged. - *Deployed note:* The separation between LIMBIC's affective output and the tool-call approval chain should be verified in `tinkerclaw-computational-humor`'s integration with security gate pathways.

11.8 THALAMUS — Curiosity Motivation

Status: DESIGN ONLY

THALAMUS is a proposed Curiosity-driven Computation Architecture (CCA) that uses LoRA fine-tuning to calibrate the agent's novelty-seeking behavior, directing autonomous exploration toward content estimated to have high information value. The CCA is unbuilt; the LoRA training process is unstarted.

Attack surface (prospective). If THALAMUS is implemented: curiosity poisoning — adversarially positioning content so that the curiosity model ranks it as high-information, luring the agent to retrieve and process attacker-controlled material it would otherwise deprioritize; LoRA weight integrity — a compromised or tampered curiosity fine-tune shifts the entire novelty-seeking function toward attacker-controlled content classes, a systemic and persistent bias rather than a single poisoned input; exploration-to-instruction escalation — content encountered during curiosity-driven exploration must not gain instruction-following authority; the transition from "retrieved for novelty" to "acted upon as instruction" is the exploitation point, and its location in the CCA pipeline requires explicit design-time attention.

AEGIS controls (design-phase). Curiosity-driven exploration must be sandboxed: content surfaced by THALAMUS's novelty signal must not gain elevated instruction-following trust by virtue of being retrieved. LoRA weight manifests must be hash-verified before each load; missing or tampered manifests should fail closed. The point in the CCA pipeline where curiosity-ranked content enters the instruction-processing pathway requires explicit trust boundary placement as a first-class design requirement.

11.9 HIVEMIND — Corporate Agent Swarm

Status: DESIGN ONLY

HIVEMIND is a proposed framework for coordinated multi-agent swarms in enterprise environments: agents share state, delegate subtasks, synchronize context, and reach swarm-level consensus on complex decisions. No implementation exists.

Attack surface (prospective). Swarm architectures create inter-agent trust relationships that expand the attack surface multiplicatively. Compromised-node propagation: a single agent whose persona or rules have been hijacked can propagate malicious state or instructions to peer agents via normal swarm communication channels, turning one compromised node into a vector for swarm-wide compromise. Cross-agent trust forgery: an attacker controlling one low-privilege agent crafts messages that appear to originate from a higher-privilege peer, escalating effective permissions without directly compromising the privileged node. Swarm consensus manipulation: Byzantine-fault scenarios where a small number of compromised agents systematically bias swarm-level decisions — the analog to the SYNAPSE deliberation-steering attack (§11.5) but operating at swarm scale and without SYNAPSE's model-diversity controls as a mitigating factor.

AEGIS controls (design-phase). HIVEMIND inter-agent messages must be cryptographically signed by the originating agent and verified before acting on claimed identity or permission scope. Byzantine fault-tolerant consensus protocols — where no coalition of N/3 compromised nodes can manipulate outcomes — should be the architectural default, not an optional hardening. Trust levels must be derived from cryptographic identity verification, not inherited from message sender claims. These controls are architectural requirements that must be designed in before any HIVEMIND implementation begins; retrofitting Byzantine-fault tolerance is substantially more costly than designing for it from the start.

11.10 AMYGDALA — Learned Intuition and Safety Gate

Status: PHASE 1 (`tinkerclaw-learned-intuition`; learned ensemble observe-only, deterministic AEGIS native enforcement floor enabled and live under bypassPermissions — see §11.14.4)

AMYGDALA operates a learned ensemble alongside a deterministic gate. The original design ascribed a *harmfulness-scoring* role to the learned half — a "Prudence" ensemble that would evaluate every tool call before execution, score it for safety risk, and block when the score crossed a threshold or when ensemble disagreement triggered a conservatism override. That role has now been measured offline, and the measurement decisively reshapes the claim. On frozen MiniLM embeddings (RTX 3080), the supervised "vicarious danger" head intended to carry harmfulness classification failed *below chance* (within-class AUROC 0.286), while two *structural* signals validated cleanly: a k-NN **novelty** detector at AUROC 0.875 and a clause-cosine **incongruity** detector at AUROC 0.896. The honest consequence — developed in §4.10.2 and §11.14.4 — is that the learned ensemble is a **novelty/incongruity signal**, not a danger classifier. It answers "is this action unlike what I have seen before, or internally incongruous?" — a triage-and-observability role (a §4.7a-adjacent signal) — and must never be counted as a harmfulness *boundary* when an adversary is present (§4.5). The Personality ensemble calibrates behavioral tone and writes nudges to a configuration file that subsequent sessions consume. The *learned* ensemble runs observe-only — it logs the novelty/incongruity verdict it would surface without aborting — while a *deterministic* AEGIS native enforcement floor (the rule-based destructive-command gate plus an AEGIS pre/post-check, registered as a PreToolUse hook) now hard-blocks unconditionally and is **enabled live under `bypassPermissions`**: the host honors the `{block}` verdict and the floor fires end-to-end. The reference-implementation audit in §11.14.4 details the current state. The boundary that actually fires is the deterministic floor, not the neural classifier — and that division is now established by measurement, not merely asserted by phase.

Attack surface. AMYGDALA is the gate in front of every tool call, making it simultaneously the most powerful safety control in the deployed architecture and the highest-value attack target. Three distinct surfaces:

1. *Learned-ensemble poisoning:* adversarial samples that shift the novelty/incongruity reference distribution so that the adversary's intended action no longer registers as anomalous — the structural detector becomes blind to exactly the pattern it should flag. (Note that this surface targets the ensemble's *triage signal*, not a harmfulness boundary; the measured collapse of the supervised danger head means the ensemble was never the boundary to begin with.) Since the ensemble's online training loop has not started (Phase 1), the training pipeline itself is a prospective attack surface for the production deployment that follows.

2. *Personality nudge-file tampering:* `~/.openclaw/cognitive/personality-nudge.json` is a writable file in the current deployment that directly influences the agent's behavioral calibration in subsequent sessions. An attacker with filesystem write access — including an attacker operating through a compromised skill process — can persistently modify agent behavior without touching any code, plugin, or configuration file that a traditional security audit would inspect. This is a live attack surface in the current Phase 1 deployment and represents the most immediately actionable AEGIS gap in the running system.

3. *ONNX model-weight substitution:* the compiled learned-ensemble models are loaded as ONNX binaries. Replacing these with adversarially modified weights — models that surface no anomaly for the specific tool calls the attacker intends — silently blinds the structural signal for the targeted action class while appearing healthy to all status checks. (Because the learned ensemble is signal, not boundary, this substitution degrades observability rather than removing a hard control; the deterministic floor of §11.14.5 is unaffected.)

AEGIS controls. - *High priority — recommended for current Phase 1 deployment:* The personality nudge file must be treated as a security-sensitive configuration artifact. File permission hardening — restricting write access to the owning agent process, not to skill processes or spawned

subagents — and hash verification at load time are AEGIS-required controls. These should be implemented before the learned ensemble's structural signal is wired into any ASK-gate that influences execution. - *High priority — recommended for current Phase 1 deployment:* ONNX model weights must be hash-verified against a known-good manifest before each load. A missing or tampered manifest should fail closed — block all tool calls — rather than fall back to unprotected execution. - *Recommended:* Amygdala Report monitoring — systematic "always green" (the learned ensemble never surfacing a novelty/incongruity flag on any tool call) is a degradation signal indicating either ensemble failure or that the relevant action class is not being evaluated. This condition should be flagged and investigated rather than treated as a sign of a clean environment. Crucially, an absence of *learned* flags is not evidence of safety — the deterministic floor (§11.14.5), not the ensemble, is the control that holds under attack.

11.11 MYELIN — Budget Prompting and Context Management

Status: CONCEPTS ADOPTED (no dedicated plugin; ideas absorbed into ENGRAM and session management)

MYELIN defines a compaction and context-budget management architecture: shrinking conversation context at defined checkpoints to stay within model context windows while preserving decision-relevant history. No MYELIN-as-plugin exists; its core concepts have been absorbed into ENGRAM v4.0 and OpenClaw's native session management infrastructure.

Attack surface. Context management creates three exploitable surfaces. Budget exhaustion attack: flooding the agent with high-volume, low-value content forces frequent compaction cycles; each compaction is an opportunity for adversarially timed erasure of context the attacker wants removed from the agent's active window — evidence of a prior injection attempt, a security warning, a flagged interaction. Cache-key collision: crafted input that produces the same cache entry as a different legitimate input causes the agent to serve a cached response — potentially a cached approval — for a request it has not actually processed. Compaction-induced evidence erasure: even benign, correctly functioning compaction can remove audit-critical context; an attacker who knows the compaction schedule can time an attack to occur immediately before a compaction cycle, ensuring the attack trace does not survive into the next session.

AEGIS controls. - *Recommended:* Security-critical events must survive compaction. Flagged interactions, tool-call denials, detected injection attempts, and audit-trail entries must be written to a persistent log before the compaction scope can include them. - *Recommended:* Compaction operations must not erase events within the ENGRAM hash-chain without preserving a compaction record — the compaction event itself must be auditable. - *Deployed note:* Because MYELIN's concepts have been absorbed into ENGRAM, ENGRAM's compaction behavior should be reviewed against these controls as part of the ENGRAM security review.

Reversible compaction is a solved engineering pattern — and AEGIS specifies the integrity properties that turn it into a *security* control. The controls above prescribe "security-critical events must survive compaction" but, in the original framing, named no concrete mechanism beyond "keep a persistent log." A production system now demonstrates the missing mechanism. chopratejas/headroom (24.7k, Apache-2.0) ships reversible **Compress-Cache-Retrieve (CCR)**: compaction never discards the original — the full content is cached and retrievable on demand, with per-content-type compressors (statistical JSON crushing, AST-aware code compression via tree-sitter, log/diff handlers) and a reproducible evaluation suite reporting 60–95% token savings at roughly zero accuracy delta on GSM8K, TruthfulQA, SQuAD, and BFCL. The security-relevant property is not the token economy — it is **reversibility by construction**: a compaction layer that caches originals and retrieves them on demand is, structurally, the compaction-survival guarantee this section requires.

The honest differentiation matters and AEGIS states it plainly. headroom is a *capability/cost* layer: its retrieval is optimized for accuracy-preserving token economy, and it is **not** designed as a tamper-evident security log — there is no hash-chaining, no write-gate, and no adversary

model, so an attacker who can write the cache can rewrite the "original." AEGIS's §11.1/§11.11 requirement is the *security* superset: the cache of originals must additionally be **append-only and hash-chained** (the §11.1 tamper-evident control) so that the retrievable original is provably the real one rather than an attacker's substitution. The positioning, therefore: *headroom proves reversible compaction is practical and cheap; AEGIS specifies the additional integrity properties that convert reversible compaction into an audit-survival security control.* This is also a clean instance of the reversibility-and-observability doctrine (§12.5): reversibility-by-construction earns the aggressive autonomy of background consolidation *provided* the categorical integrity boundary — the hash-chained append-only cache — stays hard. headroom delivers the reversibility; the hard boundary is what AEGIS adds on top.

The same positioning applies in §4.7a, where headroom is cited as the existence proof that "survive compaction" is an engineering pattern with a shipped reference, not an aspiration.

11.12 PREFRONTAL — Compounded Intelligence and Orchestration

Status: PARTIAL (`extensions/prefrontal`; recipe catalog at `extensions/tinkerclaw-prefrontal/recip`

PREFRONTAL manages multi-step autonomous execution. It maintains recipe-driven orchestration plans, dispatches subagents for parallel or sequential task execution, aggregates results, and maintains orchestration state visible to the user through the Prefrontal panel. The recipe catalog is the behavioral backbone of autonomous multi-step work.

Attack surface. PREFRONTAL presents the highest-consequence attack surface in this architecture for autonomous execution scenarios — a compromise here affects not individual tool calls but entire multi-step plans. Three specific surfaces:

1. *Recipe poisoning:* a malicious recipe file added to `extensions/tinkerclaw-prefrontal/recipes/` is auto-followed by the orchestrator when a matching trigger fires. Since recipes define multi-step execution plans that include tool calls, spawned agents, and external communications, a poisoned recipe can direct the agent through a sequence of harmful actions while appearing to follow a legitimate workflow. The trigger-matching system makes this a particularly powerful vector: the attacker does not require real-time access to the system — they need only plant a recipe that fires when the right condition occurs.

2. *Subagent dispatch hijack:* manipulation of the orchestration state causes PREFRONTAL to spawn subagents with forged instructions or inflated permission scope. A compromised orchestrating agent can use this to escalate through the subagent hierarchy without triggering individual tool-call approval gates.

3. *Recipe-state CLI spoofing:* the `recipe-state` CLI writes the status the user sees in the Prefrontal panel. An attacker who can forge recipe-state updates can misrepresent the agent's actual activity — displaying benign progress while the agent executes a different workflow — removing the user's primary real-time visibility into what is happening.

AEGIS controls. - *High priority — recommended:* Recipe files must be treated as code artifacts, not data files. Any recipe addition or modification in `recipes/` must require explicit human approval — applying the same controls used for skill installation. Auto-following of unreviewed recipes from any source other than the canonical recipes directory should be disabled by default. - *Recommended:* Subagent spawning must validate that dispatch instructions originate from an authorized orchestration context. Spawning triggered by content arriving via external inputs — email, tool results, web content — requires an explicit human approval gate. - *Recommended:* Recipe-state updates must be authenticated to prevent UI spoofing. If the recipe-state CLI accepts unsigned updates from any process, it is a deception surface that bypasses the user's primary visibility mechanism.

11.13 MNEMOSYNE — Memory Enhancements

Status: PHASE 1 (`tinkerclaw-memory-enhancements` v0.1)

MNEMOSYNE provides enhanced memory retrieval and cross-referencing capabilities extending HIPPOCAMPUS. It adds cross-reference indexing between memory segments, enabling the agent to surface associatively linked content that standard keyword retrieval would not rank highly enough to return.

Attack surface. MNEMOSYNE's attack surface substantially overlaps with HIPPOCAMPUS (§11.2) — index poisoning and concept-cluster attacks apply here directly, and the HIPPOCAMPUS controls (§11.2) are the baseline. Two Mnemosyne-specific surfaces extend beyond that: cross-reference graph manipulation — poisoning the associative links between memory segments to create false chains, causing the agent to treat unrelated memories as causally linked or to retrieve adversarially adjacent content when querying a legitimate topic; retrieval-enhancement exploit — MNEMOSYNE's enhanced retrieval surfaces content that HIPPOCAMPUS's standard retrieval would rank too low to return; an attacker who plants low-ranked malicious content knowing MNEMOSYNE will elevate it is exploiting the enhancement mechanism itself as the attack path. The more capable the retrieval, the more powerful this surface becomes.

AEGIS controls. - The core controls from §11.2 HIPPOCAMPUS apply in full: hardcoded reads for security-critical policy; retrieved content is not instruction-following authority regardless of retrieval rank or confidence. - *Additional:* Cross-reference graph integrity should be verified against the ENGRAM append-only event log — graph edges should not exist for memory segments that lack a corresponding legitimate event in the audit trail. - *Additional:* Enhanced retrieval results must not receive higher instruction-following trust than standard retrieval results. MNEMOSYNE improves recall; it must not implicitly upgrade trust level for the content it surfaces.

11.14 The Reference Implementation — What TinkerClaw Enforces Today

The preceding subsections describe attack surfaces and the controls AEGIS *recommends* for each. This subsection does something different and, for a practitioner, more useful: it audits one production agent runtime — a privately maintained OpenClaw fork that hosts the cognitive-agent plugins described throughout §11 — and reports what it *actually enforces*, control by control, with the enforcement point named. The framework is theory-forward by design; this chapter is the empirical counterweight. It doubles as a worked example of the immune-system analogy from §1.1: a **code layer** that behaves like innate immunity (fast, hard-coded, unconditional) and a **prompt layer** that behaves like adaptive immunity (learned, contextual, advisory).

The honest headline is the part most agent-security write-ups omit. The controls that block today are the deterministic ones: shell-wrapper CLIs, browser-relay confinement, a regex destructive-command gate enforced as a live PreToolUse hook, a pre-push secret scanner, and owner-scoped tools. The learned gate the framework's §4.10.2 anticipated — the ensemble once imagined to *score every tool call for harmfulness* — has now been measured, and the measurement is itself the most paper-worthy result in this chapter: its supervised harmfulness head failed below chance, so the learned half is demoted from prospective boundary to novelty/incongruity signal and remains observe-only. The lesson generalizes beyond this one runtime: in any agent system, count the deterministic controls when an adversary is present, and treat the learned ones as signal, not boundary (§4.5). What changed in the framework's favor is that the deterministic floor is no longer merely *wired* — it is enabled live under `bypassPermissions`, and the host honors its `{block}`.

11.14.1 Two layers for every control. Each control in TinkerClaw exists in one or both of two places. The **code layer** is enforced by a process the model cannot reason its way past — a shell wrapper on the executable search path, a guard in a browser extension's command handler, a git hook. The **prompt layer** is advisory text in the system prompt and behavioral-rules

files. This is the Swiss-cheese model of §12 made concrete: the prompt layer reduces the *rate* of boundary violations in non-adversarial operation; only the code layer holds when an input is actively hostile. Where a control exists only in the prompt layer, an injection that reaches the model can bypass it — which is why the AEGIS prescription (§4.5) is to promote prompt-only guards to code wherever the action is irreversible or outbound.

11.14.2 Irreversible-action gates (the "Inbox Zero" lesson, in code). §1.4 recounts the Inbox Zero incident — an agent that interpreted "clear out the old stuff" as authorization to permanently delete months of email. The structural fix is reversibility gating: a destructive request is silently downgraded to its reversible equivalent. TinkerClaw enforces this not in the prompt but in **shell-wrapper CLIs that shadow the real mail binaries on the executable search path**, so the model never reaches the irreversible operation:

- **Outlook send / reply / forward are hard-blocked.** The Outlook wrapper maintains an explicit blocklist of the Graph API paths that transmit mail (`/sendmail`, `/send`, `/reply`, `/replyAll`, `/forward`); a request to any of them throws before the network call is made. Compose paths are redirected to *draft* creation only — a reply is generated via the Graph `createReply` endpoint, which produces a threaded draft and never sends.
- **Outlook delete is rewritten to archive.** A delete request is translated into a move-to-Archive-folder operation. A real Graph `DELETE` is never issued from the wrapper, so an accidental or injection-induced deletion is structurally impossible; the operator can still delete manually in the Outlook UI, and the archived message is fully recoverable. The reversibility contract is uniform across providers — no mail-destructive operation reaches the provider from the agent, whether the channel is Outlook or Gmail.
- **Prompt-layer twin.** The behavioral-rules file carries the matching advisory rules ("reversibility gates action"; "I draft, the operator sends; no half-baked outbound to real people"). The advisory layer explains *why*; the wrapper is *what actually holds* if the model is persuaded otherwise.

This is §4.3 (programmatic guardrails) and §4.8 (self-modification prevention's sibling, irreversibility prevention) realized as a fifty-line wrapper rather than a runtime. It is also the §11.1 ENGRAM "memory write gate" principle applied to the action plane instead of the storage plane.

11.14.3 Browser-relay confinement. When the agent drives a browser, it does so through a Chrome-extension relay that mediates every Chrome DevTools Protocol command. The governing contract is the operator's: *"this is the tab I shared, not any tab the agent decides to visit."* The relay's command handler enforces three deterministic rules:

- **No new tabs or windows.** The DevTools `Target.createTarget` command is rejected outright; the agent can only act within tabs the operator has explicitly shared.
- **No cross-site navigation on shared tabs.** Before honoring a `Page.navigate`, the handler reads the tab's *current* URL live and rejects the navigation if the target is not same-site (subdomain-aware). The operator remains free to navigate their own tab anywhere; only an agent-initiated cross-site jump is blocked. The agent must follow links *within* the shared site.
- **Close limited to attached tabs.** `Target.closeTarget` is honored only for tabs the relay is already attached to.

The relay binds to loopback only (`127.0.0.1`, a fixed high port) with token authentication and attaches solely to operator-selected tabs. This is a clean instance of the §4.4 egress-control / least-privilege class: the confinement is enforced at a boundary (the extension's command handler) the agent's reasoning cannot reach, and it is precisely the kind of out-of-process enforcement §7.7's first transferable lesson recommends.

11.14.4 The gate — a live deterministic floor over a measured-but-non-boundary learned signal. The AMYGDALA component (§11.10) registers a `PreToolUse/before_tool_call` hook that evaluates every tool invocation, using an ONNX learned ensemble when the model weights are present and a deterministic rule-based gate always. An earlier framing in this program treated host-level PreToolUse blocking as a *physics limit* of the cc-bridge observe-only path — the assumption that a hook could only ever observe, never abort, because the host would not honor a block. That assumption was wrong, and the 2026-06-13 code reality resolves it: under `bypassPermissions`, a PreToolUse hook *does* deny, the host honors the `{block}` verdict, and the **AEGIS native enforcement floor is enabled and live**. Deterministic blocking is therefore real at the enforcement layer, not design-only.

Concretely: a deterministic **`AegisChecker`** is wired into the hook, and a `hard_block` verdict — raised by the rule-based destructive-command gate or by an AEGIS pre/post-check — aborts the tool call **unconditionally, independent of the observe-only trust ramp**, and that abort now fires end-to-end through the host. The distinction is exactly the deterministic/probabilistic cut of §4: the *deterministic* floor fires regardless of phase; the *learned* half is a signal, not a boundary, for the measured reason below. Two honest statements complete the picture — and a security paper that omitted them would be repeating the over-claiming this framework warns against:

1. **The deterministic gate, not a learned classifier, is the absolute veto — and that is now established by measurement, not just by deployment phase.** The learned ensemble was offline-evaluated on frozen MiniLM embeddings (RTX 3080). The supervised "vicarious danger" head — the component meant to carry harmfulness classification, the "score every tool call for safety risk" role this chapter once ascribed to it — scored **AUROC 0.286, below chance**: a frozen general-purpose embedding cannot carry harmfulness, and the within-class signal collapsed below random. Two structural detectors validated instead — a k-NN **novelty** detector at AUROC **0.875** and a clause-cosine **incongruity** detector at AUROC **0.896**. The learned half is therefore promotable only to a *novelty/incongruity ASK-gate* (a triage and observability signal), and is **not** promotable to a danger classifier — that head was measured and rejected. A second, orthogonal caveat keeps even the structural signal out of the hard boundary: the native ONNX inference dependency, while declared and configured for bundling, must be confirmed staged in the *shipped* runtime at each build — a plugin that declares a native dependency but fails to stage the platform binary falls back silently to the rule-based gate, a regression class that has bitten this runtime before. So the boundary that fires is the deterministic floor, and it fires *because* the learned path is signal — by measurement (the danger head failed below chance), not merely because the learned path is unmeasured. This is the cleanest empirical statement of the §4.5 thesis the paper has had: the supervised safety classifier was the textbook "probabilistic control treated as a boundary," and direct measurement showed it is not even a usable signal for harmfulness, only the structural anomaly detectors are.
2. The floor is only as good as its rule set (see below). It is a deterministic floor, not a complete intent verifier — the same architectural limit §7.3 identifies in NemoClaw's Layer 5.

11.14.5 The deterministic controls that block today. Independent of the learned signal, several controls enforce hard boundaries:

- **Rule-based destructive-command gate, enforced as a live PreToolUse floor.** A regex hard-block on destructive operations: recursive root deletion (`rm -rf /` and flag-order variants), filesystem formatting (`mkfs`), raw device writes (`dd of=/dev/...`, redirects to `/dev/sd*`), `chmod 777 /`, SQL `DROP` / `TRUNCATE` / unscoped `DELETE`, credential-file access (`.env`, `credentials.json`, `.aws/credentials`, `id_rsa`, `.ssh/`, `.gnupg/`), credential exfiltration over `curl`/`wget`, `git push --force` to main, and `git reset --hard`

`HEAD~N` for large N. The policy is **allow-by-default**: anything not matching a destructive pattern proceeds. This is a §4.3 control that doubles as the AEGIS pre-check feeding the §11.14.4 floor — and, as of the AEGIS native enforcement being enabled live, its `hard_block` verdict now aborts the tool call through a PreToolUse hook the host honors under `bypassPermissions`, rather than merely being logged. The honest residual: the rule set is a regex, so the same obfuscation and semantic-equivalent bypass class that limits DefenseClaw's CodeGuard (§8.4) limits this floor; it is a high-value backstop against the unsophisticated and accidental-destruction end of the distribution, not a complete intent verifier.

- **PII pre-push hook.** A git pre-push hook scans every commit about to reach the public fork for private tokens — first-name narrative use, family and location strings, host paths, business contacts, and credentials — and blocks the push on any match, while allowing the public author byline and handle. It scans two scopes (the push range *and* the accumulated drift against `origin/main`), a design that caught a real sixteen-hit leak that had accumulated across individually-clean earlier pushes. This is the §4.7b supply-chain-trust class applied to outbound code: a deterministic gate on what leaves the trust boundary.
- **Owner-only tools.** The privileged core tools — scheduling (`cron`), gateway control, and node management — reject any caller that is not the owner. This is the §4.3 permission-tier model enforced at tool dispatch.
- **Sandbox and filesystem confinement.** The runtime's published security policy constrains exec and filesystem operations to the workspace and requires sandboxing for spawned subagents. The trust model is explicitly *single trusted operator*: prompt-injection-alone, absent a demonstrated boundary bypass, is documented as out of scope. This is an honest scoping decision, not a gap — it tells an auditor exactly which threat class the runtime does and does not claim to defend, mirroring this paper's own §14.1 limitations discipline.
- **Outcome and denial logging.** Every block and every human override is written to a training ledger — the §4.7a audit substrate, and the raw material from which the learned half of the AMYGDALA gate (§11.14.4) is trained. The offline evaluation that demoted the supervised harmfulness head and validated the novelty/incongruity detectors ran against exactly this kind of accumulated live history, which is what made the structural detectors (novelty 0.875, incongruity 0.896) trainable on real data while the danger head had no reliable harmfulness signal to learn from.

11.14.6 What this reference implementation teaches. Three lessons transfer to any agent builder or operator, independent of this specific runtime:

1. **The blocking controls are deterministic and now enforced live; the learned ones are signal — and that division is measured, not assumed.** Every control that actually held under the audit was a code-layer mechanism the model could not influence, and the AEGIS native floor among them now aborts through a host-honored PreToolUse hook rather than merely logging. The learned gate's value is observability — surfacing novelty and incongruity — because the one component meant to be a *boundary* (the supervised harmfulness head) was measured at AUROC 0.286, below chance, and rejected. This is §4.5's calibration stated as an engineering measurement rather than a recommendation: a frozen-embedding safety classifier is not even a usable harmfulness *signal*, let alone a boundary.
2. **A wrapper is a security boundary; a prompt is a preference.** The reversibility contract that prevents an Inbox Zero recurrence lives in a shell wrapper, not in the system prompt — because the prompt is bypassable by the very injection class (§4.5) the wrapper is there to survive. Where an action is irreversible or outbound, the AEGIS prescription is to push the guard down to code (§11.14.2).

3. **Disclose what is enforced, what is measured, and what is still signal — by its current behavior, not its intended behavior.** The honest report now has three distinct statements rather than one hedge. (a) The deterministic floor is *enforced live* — the earlier "the host may not honor a block" framing was wrong and is corrected. (b) The learned harmfulness classifier is *measured and rejected* — not "not yet measured," but tested at 0.286 AUROC and demoted. (c) The learned structural detectors (novelty 0.875, incongruity 0.896) are *validated as signal but not promoted to a boundary*, and their native inference dependency must still be confirmed staged in the shipped runtime at each build before even the signal is counted. Reporting (b) as a result rather than a promise is strictly stronger honesty than the previous "configured, not yet measured" language — and it pre-empts the reader who would assume "measured" means "validated as a boundary." An agent-security posture that reports a control by its *intended* behavior rather than its *current* behavior is the over-claiming failure mode §4 was written to prevent.

Open upgrade items flowing from this audit, offered as the §14.2 future-work agenda made specific. One prior item is now *retired by result rather than promise*: the learned AMYGDALA ensemble has been benchmarked, and the supervised harmfulness head is rejected — so the open work is not "calibrate the danger classifier" but "operationalize the *validated* novelty/incongruity signal as an ASK-gate and observability channel, and never count it as a boundary." The remaining items: confirm at build time that the native inference dependency is actually staged into the shipped plugin runtime — declaring it is not the same as shipping it — before the structural signal is counted at all; harden the personality-nudge file (§11.10) and add load-time hash verification; adopt pairing-code / scope-intersection authentication for subagent reconnect (the §2.1.1 GHSA-hf68 mitigation, re-validating the toolset intersection on every reconnect rather than trusting the initial pairing); and pair the §1.4 CVE-2026-25253 WebSocket-hijack case study with the loopback-bind-plus-token-auth pattern the browser relay (§11.14.3) already demonstrates.

---

12. AEGIS: The Defense-in-Depth Architecture

12.1 The Swiss Cheese Model

The Swiss cheese model of safety, developed by Reason (1990) in aviation and adapted to software security, visualizes each control as a layer of cheese with holes. Individually, each layer has weaknesses. Layered together, the holes rarely align. Like airport security — the ticket check, the ID verification, the metal detector, the baggage scanner, and the air marshal in the cabin are all independent systems. Any one can fail and the others still catch the threat.

AEGIS proposes seven deterministic layers, with probabilistic controls as a general-purpose additional filter rather than a counted security layer.

The seven counted layers, in order:

1. **Network Controls** (§4.4) — port binding, egress firewall, VPN isolation.
2. **Process Isolation** (§4.2) — container, micro-VM, capability manifest, namespace.
3. **Filesystem ACLs** (§4.1) — restricted user, AppArmor/SELinux enforcing mode.
4. **Programmatic Guardrails** (§4.3) — tool allowlist, parameter restriction, permission gates, approval UX.
5. **Privacy Routing** (§4.6) — sensitivity classifier → local model for Tier 3.
6. **Configuration Immutability** (§4.8) — root-owned config, read-only agent filesystem, human-in-loop gate for self-modification.

7. **Audit and Supply-Chain Trust** (§4.7) — tamper-evident logs, anomaly detection, signed-artifact provenance, CVE scanning.

12.2 Layer Interactions

Each layer compensates for adjacent weaknesses:

- **Network controls** stop remote exploits but not locally-delivered attacks → **process isolation** limits local-attack scope.
- **Process isolation** is defeated by kernel exploits → **filesystem ACLs** at kernel level catch container escapes.
- **Filesystem ACLs** only protect listed paths → **programmatic guardrails** restrict operations even on accessible paths.
- **Programmatic guardrails** can be circumvented by creative permitted-operation sequences → **privacy routing** ensures Tier-3 data stays off external APIs regardless.
- **Privacy routing** depends on classification accuracy (5–15% false negative (Microsoft, 2024)) → **audit logging** provides visibility into routing decisions for post-hoc review.
- **Audit logging** is detection-only → **configuration immutability** ensures the audit mechanism cannot be disabled before detection.
- **Configuration immutability** does not cover inputs the agent already trusts → **supply-chain trust** (signed skills, SLSA provenance, sleeper-agent-aware model governance (Hubinger et al., 2024)) validates what reaches the immutable configuration in the first place.

12.3 Worked Attack Simulation

We trace twelve representative attack scenarios through a Standard Configuration AEGIS deployment. A cell marked **BLOCKED** indicates the scenario is stopped at that layer; **DETECTED** indicates the scenario passes physical prevention but is visible in audit for response; **pass** indicates the layer does not address that class.

#	Attack Scenario	L1 Net	L2 Proc	L3 FS	L4 Guard	L5 Priv	L6 Immut	L7 Audit/SC	Result
1	CVE-2026-25253-class WebSocket hijack (NVD, 2026)	BLOCKED	—	—	—	—	—	—	BLOCKED AT L1
2	Malicious skill exfiltrates SSH keys	pass	BLOCKED	—	—	—	—	—	BLOCKED AT L2
3	Prompt injection → read credential file	pass	pass	BLOCKED	—	—	—	—	BLOCKED AT L3

#	Attack Scenario	L1 Net	L2 Proc	L3 FS	L4 Guard	L5 Priv	L6 Immut	L7 Audit/SC	Result
4	Allowed tool chain → arbitrary exec via `git` hook	pass	pass	pass	BLOCKED	—	—	—	BLOCKED AT L4
5	Client PII sent to frontier LLM API	pass	pass	pass	pass	BLOCKED	—	—	BLOCKED AT L5
6	Agent attempts to modify SOUL.md	pass	pass	pass	pass	pass	BLOCKED	—	BLOCKED AT L6
7	Credential exfil via DNS tunneling (Cisco Talos, 2024)	pass	pass	pass	pass	pass	pass	DETECTED	DETECTED AT L7
8	Delayed-action memory injection (Greshake et al., 2023)	pass	pass	pass	pass	pass	pass	DETECTED	DETECTED AT L7
9	User rubber-stamps irreversible action	pass	pass	pass	pass*	pass	pass	DETECTED	DETECTED AT L7
10	Subagent reconnect privilege escalation	pass	BLOCKED†	—	—	—	—	—	BLOCKED AT L2
11	Indirect prompt injection, multi-step (Greshake et al., 2023; Zou et al., 2023)	pass	pass	pass	pass	PARTIAL	pass	DETECTED	PARTIAL (L5/L7)

#	Attack Scenario	L1 Net	L2 Proc	L3 FS	L4 Guard	L5 Priv	L6 Immut	L7 Audit/SC	Result
12	Compromised local model (sleeper / poisoned fine-tune) exfiltrates processed data (Hubinger et al., 2024)	pass	pass	pass	pass	pass	pass	pass‡	GAP — No Layer

Results: 7 fully prevented, 4 detected (enabling response), 1 partial, 1 genuine gap.

*Scenario 9: With time-delayed execution for irreversible actions (§4.3.1), this can be upgraded from DETECTED to BLOCKED AT L4.

†Scenario 10: Requires correct per-agent RBAC configuration (NemoClaw) or explicit non-inheritance policy; with default OpenClaw configuration, this passes Layer 2 and must be caught further down.

‡Scenario 12 represents a genuine architectural gap. If the local model used for privacy routing is itself compromised — via poisoned fine-tune, model-weight tampering, or a sleeper-agent backdoor (Hubinger et al., 2024) — it can exfiltrate data it processes to any endpoint accessible within the container. Mitigation requires *model integrity verification* (hash or signature of model weights against a trusted source), *outbound network restrictions within the local model container* that mirror the host-level egress policy, and — for the sleeper-agent subclass specifically — model-provenance and provider-level governance outside AEGIS's runtime scope. This class is the clearest case for the Emerging Classes in §4.10.2 (I/O content classifiers applied to the local-model output boundary) and §4.10.5 (identity hygiene applied to which entity is allowed to load the model weights in the first place).

12.4 Reference Configurations

One principle before configurations. Removing access is more effective than hardening access. When an asset does not need to be in the agent's reachable set, scope reduction is the highest-leverage action — eliminating risk rather than mitigating it.

Minimal Configuration (20 minutes). Addresses the highest-probability attack class (network exploitation) and the highest-consequence vulnerability (credential file access):

```
STEP 1: Bind agent control port to localhost only; add auth-token
        requirement for all local-origin WebSocket upgrades.
STEP 2: Verify all installed skills against known sources;
        remove unrecognized ones.
STEP 3: Apply filesystem deny policy for: credential directories,
        secret files, crypto material.
STEP 4: Set agent workspace root to an isolated subdirectory.
```

This configuration blocks Scenario 1 (CVE class) and Scenario 2 (filesystem access to credentials).

Standard Configuration (2–4 hours). Applies the full 7-layer AEGIS stack in forms accessible to technically competent individuals without enterprise infrastructure:

```
STEP 5:  Create dedicated low-privilege user account for agent process.
STEP 6:  Run agent in a rootless container with a seccomp-bpf profile;
         mount workspace subdirectory only.
STEP 7:  Egress firewall: permit LLM API IPs only; deny all other
         outbound; enforce via sidecar proxy where possible.
STEP 8:  Privacy routing: classify requests; route Tier 3 content
         to local model.
STEP 9:  Root-owned, read-only behavioral configuration files;
         human-in-loop gate for Level-2 self-modifications.
STEP 10: Daily security scan: skill manifest changes, permission drift,
         CVE status.
STEP 11: Execute DPA with LLM provider(s) before processing any
         third-party PII.
```

Enterprise Configuration (formal security review + ongoing governance). All of Standard, plus:

```
STEP 12: Out-of-process policy enforcement (OpenShell/equivalent).
STEP 13: Per-session isolation for multi-agent workflows;
         short-lived workload identities (SPIFFE/SPIRE) per session.
STEP 14: Enterprise DLP integration for data classification.
STEP 15: Signed skill manifests with organizational PKI;
         SLSA provenance tracking for model artifacts.
STEP 16: Tamper-evident audit trail integrated with SIEM.
STEP 17: IT security review and change management documentation
         mapped to NIST AI RMF.
STEP 18: Penetration testing of agent deployment.
STEP 19: Employee prompt-injection awareness training.
STEP 20: Incident response plan specific to agent compromise,
         with explicit sleeper-agent triage playbook.
```

12.5 Reversibility-and-Observability as a Safety Control Class

The seven counted layers of §12.1 are framed as barriers — controls that *prevent* a harmful action from completing. This subsection adds a complementary safety strategy that does not prevent the action but makes it *safe to permit*: a control that lets the agent act freely while guaranteeing the action is **reversible by construction** and **fully observed**. It is the framework-level answer to a question §4.5, §4.8, and §11.14 circle around but never state crisply: *when is it safe to remove a gate?* The answer this class gives: remove the gate exactly when the action is (a) reversible by construction and (b) fully observed — and never for the categorical security boundaries.

The doctrine is *default to maximum capability and autonomy; make it safe with reversibility plus observability, not with a permission gate; loosen as confidence grows.* It is the fourth trilemma position of §1.2, and it is not a relaxation of AEGIS's deterministic discipline — it sharpens it, via an explicit **two-axis cut** that disciplines the otherwise-vague "be less restrictive" intuition:

1. **Quantity bounds are derived, never frozen — soft pressure, not a cliff.** Timeouts, retry counts, loop/recursion caps, concurrency width, and token/tool-call budgets should be computed from the live situation (remaining allowance, time-to-reset, value of the work in flight) and apply graduated back-pressure *before* any ceiling. Stated as a bug class: *a hard quantity threshold that can kill a near-done task or amputate a capability is*

a defect; a frozen number is at most a safety CEILING, never the working value. The shipped reference is concrete — per-spawn token and tool-call budgets are validated and normalized at the RPC boundary (a negative or non-integer budget is rejected as a client error), but the *working* value is derived per spawn from remaining budget and fan-out, and an unthreaded budget leaves the clamp inert at infinity rather than defaulting to an arbitrary cap. Error-recovery retry counts and supervision-loop counts are derived the same way.

2. **Categorical capability/permission boundaries stay hard — deterministic, unconditional.** The PII public/private split, the destructive-command regex gate (§11.14.5), owner-only privileged tools, and per-spawn tool whitelists are *not* on the soft axis; they remain absolute, model-unreachable boundaries. This is the §4 deterministic/probabilistic cut re-expressed as a deterministic/*derived* cut applied *within* the controls AEGIS classifies as deterministic: a security boundary is categorical and hard; a resource *bound* is quantitative and derived. Conflating the two — treating a token budget as a security perimeter, or a security perimeter as a tunable knob — is a named error.

The class strengthens three sections concretely. **§4.8 (self-modification):** reversible-and-observed mutation — an append-only never-delete archive plus a live-margin promotion bar — is presented alongside configuration immutability as a legitimate, arguably stronger, defense against the agent-as-its-own-adversary; the choice between them is a §12.5 decision, sound only because the integrity boundary (hash-chained append-only archive) stays hard. **§11.11 (compaction):** reversible CCR compaction (headroom) is precisely a case where reversibility-by-construction earns the aggressive autonomy of background consolidation, *provided* the hash-chained append-only cache stays hard. **§14.2 (future work):** the observe-only AMYGDALA ledger (§11.14.5) is the empirical lever that makes "loosen as confidence grows" *measurable* — observability is the prerequisite that converts the autonomy-first posture from faith into evidence-gated graduation. The honesty discipline of §11.14 carries through unchanged: this doctrine is sound only because axis (2) stays hard. An agent system that softened its security perimeters under the same banner would be over-claiming in exactly the way §4 warns against.

---

13. Per-Persona Recommendations

13.1 Persona A — The Tinkerer

Immediate (20 min): Bind control port to localhost. AppArmor deny on credential directories and crypto seed files. Verify installed skills.

Standard (3 hours): Container with workspace-only mount. Egress firewall. Root-owned configuration files. Cron security scan.

Goal configuration: Trilemma position — High Capability + High Autonomy + Moderate Safety. The investment is proportionate to the data at risk (primarily financial accounts and personal credentials).

13.2 Persona B — The Freelancer

Immediate (1 hour): Bind control port. Filesystem deny on client directories except explicit workspace. Execute DPA with LLM provider(s). Restrict email access to specific non-privileged inboxes.

Standard (4–6 hours): All of Tinkerer Standard, plus: privacy routing for client data directories to local Ollama model; per-session confirmation gate on all external communications; client data in isolated workspace excluded from general agent context; quarterly scope review.

Goal configuration: Trilemma position — High Capability + Moderate Autonomy + High Safety. The "moderate autonomy" means human gate on all external actions (email sends, file writes outside workspace, API calls to non-LLM endpoints).

Regulatory baseline: GDPR data inventory mapping which client PII the agent can access; DPA executed with each LLM provider; client-consent policy for AI-assisted work.

13.3 Persona C — The SMB Operator

Immediate (2 hours): Scope restriction before hardening. Remove payroll and HR data from agent-accessible paths. Execute DPA with LLM provider. Inform staff of agent's capabilities and data scope.

Standard (1 day): Container isolation with specific workspace per task type (financial workspace, client workspace, operations workspace — each separately scoped). Egress firewall. Privacy routing for employee and client data. Monthly skill audit. AI BOM generated by DefenseClaw's pre-installation scanner.

Goal configuration: Trilemma position — Moderate Capability + Moderate Autonomy + High Safety. The SMB context does not have enterprise resources, but it has enterprise data obligations (employee PII, client data). The right response is to genuinely restrict what the agent can access rather than trying to harden an overly broad access scope.

Key difference from Tinkerer: The SMB Operator is a data controller for third-party data (employees, clients). Regulatory obligations apply regardless of how the organization describes its AI tool use. The DPA execution is non-optional.

13.4 Persona D — The Corporate Employee

Immediate: This configuration requires organizational response, not personal hardening. Cameron should escalate to IT security. The minimum viable organizational response:

1. Restrict agent's accessible data to Cameron's personal work files (remove production credentials, corporate network shares, privileged email)
2. Document the deployment in the IT asset register
3. Apply egress firewall restricting agent to approved LLM API endpoints only
4. Assess whether any regulatory notification is required (has any Tier 3 data already been sent to external LLM APIs?)

The uncomfortable truth for Persona D: Technical hardening cannot make an unapproved, undocumented agent deployment with access to production systems and M&A communications "acceptable." The risk profile requires organizational governance first, then technical configuration. If Cameron's organization will not approve a properly governed agent deployment, the agent should be removed from corporate infrastructure.

Goal configuration: Trilemma position — Moderate Capability + Low Autonomy + High Safety. Human approval required for all material actions. Scope limited to explicitly approved data categories.

---

14. Limitations and Future Work

14.1 Limitations

No empirical validation. AEGIS has not been tested against real attack campaigns at scale. The worked simulation (§12.3) is a structured thought experiment grounded in known attack patterns, not an empirical measurement. Future work should include controlled red-team exercises against AEGIS-configured deployments across all four personas.

Qualitative risk scores. Despite the calibration methodology in §6, probability and severity ratings remain expert judgment. The risk matrices should be treated as prioritization heuristics, not actuarial precision. Different practitioners with different threat intelligence would assign different scores.

NemoClaw and DefenseClaw assessment is snapshot-based. Both frameworks are actively developed. The gap analysis in §7 and §8 reflects April 2026 documentation. Specific gaps may be addressed in future releases.

Regex-bypass problem not fully solved. DefenseClaw's CodeGuard and the broader static analysis approach to supply chain security has known bypass vectors (obfuscation, encoding, runtime code generation). The AEGIS framework currently treats supply chain monitoring as a 3/5 control — this may be generous for sophisticated adversaries.

The liability analysis is jurisdiction-dependent. §3 provides a GDPR-centric analysis. Other jurisdictions (CCPA in California, PIPL in China, APRA in Australia) have different controller/processor concepts and different liability frameworks. Organizations operating across jurisdictions require jurisdiction-specific legal analysis, not AEGIS's framework alone.

SMB Operator persona (§5.3) is less validated than the others. The Tinkerer, Freelancer, and Corporate Employee personas have been reviewed against incident reports and regulatory guidance. The SMB Operator persona has less empirical grounding and rests more heavily on inference from analogous SME data-protection cases.

14.2 Future Work

1. **Red-team validation across AEGIS configurations.** Run controlled adversarial exercises against Minimal, Standard, and Enterprise AEGIS profiles for all four personas. Measure actual attack success rates rather than theoretical layer analysis.

2. **Liability framework validation.** Engage legal counsel in relevant jurisdictions to validate the controller/processor analysis in §3 and the DPA recommendations.

3. **NemoClaw Intent Verification improvements.** The Layer 5 weakness (evaluates proposed action type, not payload content) is the most analytically interesting gap. Research into LLM-based intent verification that operates at payload content level, not action type level, would significantly improve the weakest NemoClaw layer.

4. **SMB configuration tooling.** The Standard Configuration in §12.4 is achievable but requires technical competence that not all SMB Operators have. A configuration wizard or audit tool that walks through the Standard Configuration steps would substantially improve SMB security posture.

5. **Self-modification threat model formalization.** The CEREBELLUM attack surface (§11.4) requires more formal treatment. A threat model specifically for self-modifying agent systems — analogous to formal verification for self-modifying code — would strengthen AEGIS's coverage of this scenario.

6. **Memory integrity verification.** Extending AEGIS to address the temporal attack surface of memory poisoning with cryptographic integrity guarantees for the event store

would close the gap between ENGRAM's design and its security properties. The reversible-compaction reference (headroom, §11.11) supplies the retrieval half; the open work is hash-chaining the cache of originals so the retrievable original is provably authentic.

7. **Operationalizing the validated novelty/incongruity signal.** The learned safety gate has now been measured (§4.10.2, §11.14.4): the supervised harmfulness head is rejected, but the structural novelty (0.875) and incongruity (0.896) detectors validated. The open work is no longer "calibrate a danger classifier" — that path is closed by result — but to wire the validated structural signal into an ASK-gate and observability channel without ever counting it as a boundary, and to confirm at build time that its native inference dependency is actually staged in the shipped runtime.

8. **Evidence-gated autonomy graduation.** The §12.5 reversibility-and-observability class proposes loosening controls "as confidence grows." Formalizing the graduation criterion — what observed evidence, accumulated over what window, justifies removing a gate for a reversible action class — would turn the autonomy-first posture from a doctrine into a measurable promotion procedure, using the observe-only ledger (§11.14.5) as the evidence source.

---

15. Related Work

OWASP Top 10 for LLM Applications (2025) provides a vulnerability-focused taxonomy covering prompt injection, insecure output handling, training data poisoning, and supply chain risks. AEGIS differs in focusing specifically on *agentic* systems with tool use, action execution, and state persistence, and in providing a layered defense architecture rather than a vulnerability list. The OWASP taxonomy is complementary; AEGIS is agent-specific where OWASP is LLM-general.

MITRE ATLAS (2025 refresh) is the AI- and ML-specific adversary knowledge base — the AI-domain counterpart to MITRE's general-purpose ATT&CK framework — enumerating attacker tactics and techniques against machine-learning systems. AEGIS uses STRIDE rather than ATLAS as its primary threat-modeling lens because STRIDE maps more directly to the *defensive* strategy classes proposed; ATLAS's offensive technique enumeration is complementary but serves a different analytical purpose, and the §10.4 coverage-mapping work is the natural point to reconcile the two.

NIST AI Risk Management Framework (AI RMF, 2023) provides governance-level guidance for AI system risk management. AEGIS is more technically specific, providing implementation-level security controls rather than organizational governance process guidance. The AI RMF is the governance framework above which AEGIS operates as the technical control framework.

Perez and Ribeiro (2022) on prompt injection attacks established the foundational understanding of why prompt-level controls are insufficient against adversarial inputs. AEGIS builds on this by providing the deterministic/probabilistic classification framework and concrete architectural alternatives.

Google's Secure AI Framework (SAIF, 2023) identifies six core elements of AI security. AEGIS's eight strategy classes intersect with SAIF's framework at most points, with AEGIS providing more operational specificity for agent deployments.

Reason (1990) originated the Swiss cheese model in the context of human-caused accidents in aviation. AEGIS's adaptation to agent security preserves the core insight — layered imperfect

controls achieve what no single control can — while mapping it to a software-specific control hierarchy.

chopratejas/headroom (2026) is a production open-source (Apache-2.0) context-management system implementing reversible Compress-Cache-Retrieve (CCR): compaction caches originals and retrieves them on demand rather than discarding them, with per-content-type compressors and a reproducible eval suite reporting 60–95% token savings at near-zero accuracy delta. It is the closest existence proof for the "security-critical events must survive compaction" requirement AEGIS states for ENGRAM (§11.1) and MYELIN (§11.11). The relationship is capability-vs-security: headroom optimizes reversible compaction for accuracy-preserving token economy and explicitly does *not* provide a tamper-evident security log (no hash-chaining, no write-gate, no adversary model). AEGIS specifies the integrity superset — an append-only, hash-chained cache of originals — that turns reversible compaction into an audit-survival security control. We cite headroom as the engineering reference that "survive compaction" is solved and cheap, and position AEGIS as the layer that adds the integrity boundary on top.

addyosmani/agent-skills (2026) is an open-source agent-discipline skill collection that includes **doubt-driven-development** (CLAIM → EXTRACT artifact with the proposer's reasoning stripped → DOUBT via a fresh-context adversarial reviewer → RECONCILE → STOP). It is a concrete published instantiation of the abstract "separate adversarial review pass" that AEGIS prescribes for CEREBELLUM self-modification (§11.4) and the skeptical-minority veto for SYNAPSE deliberation (§11.5). The non-obvious and transferable contribution is the EXTRACT-strips-reasoning + fresh-context-reviewer design: it defeats the failure mode in which the reviewer is anchored by the proposer's own justification. AEGIS adopts this property directly as the recommended form of its review-pass controls.

These two systems are deliberately treated as first-class related work rather than footnotes: as actively maintained open-source projects they are fresher than the static citations above and supply running references — reversible compaction and reasoning-stripped adversarial review — for controls the framework previously described only abstractly.

No prior work to our knowledge provides an integrated security framework for autonomous AI agents that spans from OS-level controls to behavioral rules, includes formal persona-based risk analysis, incorporates a data classification-based liability framework, and provides a concrete defense-in-depth architecture with worked simulation. AEGIS aims to fill this gap.

---

16. References

1. Addy Osmani / agent-skills contributors. (2026). *agent-skills: Discipline Skills for Coding Agents* (incl. doubt-driven-development). Open-source skill collection, GitHub `addyosmani/agent-skills`. https://github.com/addyosmani/agent-skills

2. Agache, A., Brooker, M., Iordache, A., Liguori, A., Neugebauer, R., Piwonka, P., & Popa, D.-M. (2020). Firecracker: Lightweight Virtualization for Serverless Applications. *USENIX NSDI 2020*. https://www.usenix.org/conference/nsdi20/presentation/agache

3. Anthropic. (2024). Developing a Computer Use Model. *Anthropic Engineering Blog*, October 2024. https://www.anthropic.com/news/developing-computer-use

4. Anthropic. (2025). *Anthropic Data Processing Addendum for Business Customers*. Anthropic Technical Documentation.

5. Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., et al. (2022). Constitutional AI: Harmlessness from AI Feedback. *arXiv:2212.08073*. https://arxiv.org/abs/2212.08073

6. Berman, M. (2026). "I had a hacker try to break into my AI agent — here's what happened." YouTube, Berman AI channel, February 2026.

7. Center for Internet Security. (2024). *CIS Benchmarks for Linux Distributions* (2024 editions). https://www.cisecurity.org/cis-benchmarks

8. Cisco Systems. (2026). "DefenseClaw: Open-Source Security for OpenClaw Deployments." *Cisco Developer Blog*, March 27, 2026.

9. Cisco Talos Intelligence Group. (2024). DNS-over-HTTPS Abuse for Exfiltration and Command-and-Control. *Talos Blog*, 2024. https://blog.talosintelligence.com/

10. Cisco Talos Intelligence Group. (2025). Malicious OpenClaw Skills: A Supply Chain Attack Analysis. *Talos Intelligence Blog*, October 2025. https://blog.talosintelligence.com/

11. Chopra, T. / headroom contributors. (2026). *headroom: Reversible Compress-Cache-Retrieve Context Management for LLM Agents*. Apache-2.0, GitHub `chopratejas/headroom`. https://github.com/chopratejas/headroom

12. Cloud Security Alliance. (2025). *AI Controls Matrix v1.0*. CSA Working Group on AI Controls. https://cloudsecurityalliance.org/research/ai-controls-matrix

13. Cowan, C., Beattie, S., Kroah-Hartman, G., Pu, C., Wagle, P., & Gligor, V. (2000). SubDomain: Parsimonious Server Security (ancestor of AppArmor). *USENIX LISA 2000*. https://www.usenix.org/legacy/events/lisa2000/full_papers/cowan/cowan.pdf

14. Debenedetti, E., et al. (2024–2025). AgentDojo: A Dynamic Environment to Evaluate Prompt-Injection Attacks and Defenses for LLM Agents. *NeurIPS Datasets and Benchmarks*. https://arxiv.org/abs/2406.13352

15. Felt, A. P., Ha, E., Egelman, S., Haney, A., Chin, E., & Wagner, D. (2012). Android Permissions: User Attention, Comprehension, and Behavior. *SOUPS 2012*. https://doi.org/10.1145/2335356.2335

16. Google. (2023). *Google's Secure AI Framework (SAIF)*. Google Safety Engineering Center, 2023–2025 updates. https://safety.google/cybersecurity-advancements/saif/

17. Greshake, K., Abdelnabi, S., Mishra, S., Endres, C., Holz, T., & Fritz, M. (2023). Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. *ACM AISec 2023*. https://arxiv.org/abs/2302.12173

18. Huang, J. (2026). Keynote Address, GTC 2026. NVIDIA Corporation, March 2026. https://www.nvidia.com/gtc/keynote/

19. Hubinger, E., Denison, C., Mu, J., Lambert, M., Tong, M., MacDiarmid, M., et al. (2024). Sleeper Agents: Training Deceptive LLMs that Persist Through Safety Training. *arXiv:2401.05566*. https://arxiv.org/abs/2401.05566

20. Jackson, C., Barth, A., Bortz, A., Shao, W., & Boneh, D. (2007). Protecting Browsers from DNS Rebinding Attacks. *ACM CCS 2007*. https://doi.org/10.1145/1315245.1315298

21. Mazeika, M., Phan, L., Yin, X., et al. (2024). HarmBench: A Standardized Evaluation Framework for Automated Red Teaming and Robust Refusal. *ICML 2024*. https://arxiv.org/abs/2402.0424

22. Microsoft. (2024). *Presidio: Context-Aware, Pluggable, and Customizable Data Protection and De-Identification SDK*. https://microsoft.github.io/presidio/

23. MITRE. (2025). *MITRE ATLAS: Adversarial Threat Landscape for Artificial-Intelligence Systems* (2025 refresh). MITRE Corporation. https://atlas.mitre.org/

24. Mosier, K. L., & Skitka, L. J. (1996). Human Decision Makers and Automated Decision Aids: Made for Each Other? In *Automation and Human Performance: Theory and Applications* (pp. 201–220).

25. National Institute of Standards and Technology. (2020). *Security and Privacy Controls for Information Systems and Organizations*. NIST SP 800-53 Rev. 5. https://doi.org/10.6028/NIST.SP.800-53r5

26. National Institute of Standards and Technology. (2023). *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*. NIST AI 100-1.

27. National Institute of Standards and Technology. (2024a). *Artificial Intelligence Risk Management Framework: Generative AI Profile*. NIST AI 600-1. https://doi.org/10.6028/NIST.AI.600-1

28. National Institute of Standards and Technology. (2024b). *Secure Software Development Practices for Generative AI and Dual-Use Foundation Models*. NIST SP 800-218A. https://doi.org/10.6028/NIST.SP.800-218A.ipd

29. National Vulnerability Database. (2026). CVE-2026-25253: OpenClaw WebSocket Origin Validation Bypass — Unauthenticated Remote Tool Injection. February 2026. https://nvd.nist.gov/vuln/detail/CVE-2026-25253

30. National Vulnerability Database / GitHub Security Advisories. (2026). GHSA-hf68-49fm-59cq: OpenClaw Agent Reconnect Scope Escalation — Critical Privilege Escalation. March 2026.

31. NVIDIA Corporation. (2026). *NemoClaw Security Architecture Reference Guide v1.0*. NVIDIA Developer Documentation, GTC 2026. https://developer.nvidia.com/

32. OpenAI. (2025). *OpenAI Data Processing Addendum*. OpenAI Business Terms.

33. Open Source Security Foundation. (2023). *Supply-chain Levels for Software Artifacts (SLSA) v1.0*. Linux Foundation / OpenSSF. https://slsa.dev/

34. OWASP. (2025). *OWASP Top 10 for Large Language Model Applications v2.0*. Open Worldwide Application Security Project. https://genai.owasp.org/

35. Parasuraman, R., & Riley, V. (1997). Humans and Automation: Use, Misuse, Disuse, Abuse. *Human Factors*, 39(2), 230–253. https://doi.org/10.1518/001872097778543886

36. Perez, E., Huang, S., Song, F., Cai, T., Ring, R., Aslanides, J., et al. (2022). Red Teaming Language Models with Language Models. *EMNLP 2022*. https://arxiv.org/abs/2202.03286

37. Perez, F., & Ribeiro, I. (2022). Ignore Previous Prompt: Attack Techniques for Language Models. *Workshop on Trustworthy NLP, NeurIPS 2022*. https://arxiv.org/abs/2211.09527

38. Pivot Point Security. (2026). MCP Server OAuth Authentication Vulnerabilities: A Survey of a 43% Exposure Rate. *Pivot Point Security Research*, March 2026.

39. Reason, J. (1990). *Human Error*. Cambridge University Press.

40. Reeder, R. W., Felt, A. P., Consolvo, S., Malkin, N., Thompson, C., & Egelman, S. (2018). An Experience Sampling Study of User Reactions to Browser Warnings in the Field. *ACM CHI 2018*. https://doi.org/10.1145/3173574.3174086

41. Russell, S. (2019). *Human Compatible: Artificial Intelligence and the Problem of Control*. Viking.

42. Schulhoff, S., Pinto, J., Khan, A., Bouchard, L.-F., Si, C., Anati, S., et al. (2023). Ignore This Title and HackAPrompt: Exposing Systemic Vulnerabilities of LLMs Through a Global Prompt Hacking Competition. *EMNLP 2023*. https://arxiv.org/abs/2311.16119

43. Shadowserver Foundation. (2026). CVE-2026-25253 Patch Adoption: 30-Day Follow-Up Survey. *Shadowserver Security Blog*, March 2026. https://www.shadowserver.org/news/

44. Smalley, S., Vance, C., & Salamon, W. (2001). Implementing SELinux as a Linux Security Module. *NSA Technical Report #01-043*. https://www.cs.unibo.it/~sacerdot/doc/papers/selinux.pdf

45. Snyk Security Research Team. (2023). DNS Rebinding Attacks on Localhost Services. *Snyk Research Blog*, 2023–2024. https://snyk.io/blog/dns-rebinding-attacks/

46. Stamper, T. (2026). Meta Internal Security Advisory: OpenClaw on Corporate Devices. Surfaced via the information-security community, January 2026.

47. Wallace, E., Xiao, K., Leike, R., Weng, L., Heidecke, J., & Beutel, A. (2024). The Instruction Hierarchy: Training LLMs to Prioritize Privileged Instructions. *arXiv:2404.13208*. https://arxiv.org/abs/2404.13208

48. Yan, M., Fletcher, C. W., & Torrellas, J. (2020). Cache Telepathy: Leveraging Shared Resource Attacks to Learn DNN Architectures. *USENIX Security 2020*. https://www.usenix.org/conference/u

49. Young, E. G., Zhu, P., Caraza-Harter, T., Arpaci-Dusseau, A. C., & Arpaci-Dusseau, R. H. (2019). The True Cost of Containing: A gVisor Case Study. *USENIX HotCloud 2019*. https://www.usenix.org/conference/hotcloud19/presentation/young

50. Zeng, Y., Yang, B., Liu, C., Kwok, J. T., et al. (2024). Exploiting Memory Editing in Language Models: Attack Surfaces and Defenses. *arXiv preprint* (representative of the 2024 memory-edit-attack literature).

51. Zou, A., Wang, Z., Carlini, N., Nasr, M., Kolter, J. Z., & Fredrikson, M. (2023). Universal and Transferable Adversarial Attacks on Aligned Language Models. *arXiv:2307.15043*. https://arxiv.org/abs/2307.15043

52. Zverev, E., Abdelnabi, S., Fritz, M., & Lampert, C. (2024). Can LLMs Separate Instructions From Data? And What Do We Even Mean By That? *EMNLP 2024*. https://arxiv.org/abs/2403.06833

Appendix A: AEGIS Quick-Reference Checklist

Universal Minimum (20 minutes — Every Deployment)

- ☐ Bind agent control port to localhost (eliminates CVE-2026-25253 class)
- ☐ Verify all installed skills against known sources; remove unrecognized ones
- ☐ Apply filesystem deny policy for credential directories and secret files
- ☐ Restrict agent workspace root to an isolated subdirectory

Persona A — The Tinkerer — Standard

- ☐ Container isolation with workspace-only bind mount
- ☐ Egress firewall: LLM API endpoints only
- ☐ Root-owned, read-only behavioral configuration files
- ☐ Cron security scan: skill changes, CVE status, permission drift

Persona B — The Freelancer — Standard

All of Persona A Standard, plus: - [ ] Execute DPA with each LLM provider before processing client data - [ ] Privacy routing: client directories → local model (Ollama or equivalent) - [ ] Confirmation gate on all external communications (email, API calls) - [ ] Client data in isolated workspace, excluded from general agent context - [ ] GDPR data inventory mapping accessible client PII - [ ] Quarterly scope audit: does the agent actually need everything it can access?

Persona C — The SMB Operator — Standard

All of Persona A Standard, plus: - [ ] Scope restriction FIRST: remove employee payroll/HR from agent-accessible paths - [ ] Execute DPA with LLM provider before *any* employee or client data processing - [ ] Separate workspace containers per data category (finance, client, operations) - [ ] Privacy routing for employee and client data - [ ] Inform staff of agent capabilities and data scope (documented communication) - [ ] Run DefenseClaw skill-scanner + CodeGuard on all installed skills - [ ] Generate AI BOM for compliance documentation - [ ] Monthly skill audit

Persona D — The Corporate Employee — Prerequisites

- ☐ **BEFORE HARDENING:** Restrict scope to personal work files only — remove all corporate network shares, production credentials, privileged email
- ☐ Escalate to IT security for formal documentation in asset register
- ☐ Assess whether any Tier 3 data has already been sent to external LLM APIs; if yes, assess regulatory notification obligations
- ☐ If organization approves continued use: apply Standard Configuration plus Enterprise egress and audit controls
- ☐ If organization does not approve: remove agent from corporate infrastructure

Enterprise Additional Controls

All applicable persona Standard, plus: - [ ] Out-of-process policy enforcement (OpenShell or equivalent) - [ ] Per-session isolation for multi-agent workflows - [ ] Enterprise DLP integration for data classification - [ ] Signed skill manifests with organizational PKI - [ ] Tamper-evident audit trail integrated with SIEM (Splunk or equivalent) - [ ] Penetration testing of agent deployment (include in scheduled pen test scope) - [ ] Employee prompt injection awareness training - [ ] Incident response plan specifically covering agent compromise scenarios - [ ] NemoClaw evaluation for organizations requiring full 5-layer architecture

---

Appendix B: Threat-to-Section Cross-Reference

For IT contractors who want to navigate directly to specific concerns:

Concern	Section
“Can a hacker take over the agent?”	§2.1 — Class I threat taxonomy
“What data is actually at risk?”	§3.1 — Three-tier data classification
“Who’s liable if Anthropic has a breach?”	§3.3–3.5 — GDPR liability analysis
“Can we recover money from Anthropic?”	§3.5 — Recovery analysis

Concern	Section
“Does Anthropic train on our data?”	§3.4 — ToS and DPA analysis
“Is NemoClaw actually better?”	§7 — Full NemoClaw analysis with gaps
“What does DefenseClaw actually do?”	§8 — Full DefenseClaw analysis
“If these tools are secure, why are NVIDIA and Cisco doing this?”	§9 — Business motivation analysis
“Has anyone actually tried to hack a real setup?”	§10 — Berman penetration test
“What does a real agent runtime actually enforce?”	§11.14 — TinkerClaw reference-implementation audit
“Code controls or just prompt rules?”	§11.14.1, §11.14.5 — code layer vs. prompt layer
“What’s the minimum I need to do right now?”	Appendix A — Quick-reference checklist
“What are the specifics for my deployment context?”	§13 — Per-persona recommendations

---

AEGIS: A Multi-Layered Security Framework for Autonomous AI Agents Author: Oscar Serra | Date: June 2026 | Version: 2.9 Classification: Internal — IT Contractor Review

References