VIRTUS CYBERSECURITY

Instruction-vs-Data Confusion as Pedagogical Spine

A Curriculum Bridging Classical Memory-Corruption to Agentic LLM Security

By **Jon Munson**
Virtus Cybersecurity

virtuscybersecurity.com

jon@virtuscybersecurity.com

April 2026

Contents

Abstract

The instruction/data confusion that produced fifty years of memory-corruption vulnerabilities is reappearing as the defining security problem of agentic AI, in a semantically-weakened form. This report argues that the classical systems-security canon provides a pedagogical spine on which to build a complete agentic-security curriculum. The relevant classical sources are Saltzer & Schroeder (1975), Hardy (1988), Dennis & Van Horn (1966), Denning (1976), Miller (2006), and Abadi et al. (2005). The architectural consensus emerging in 2024-2026 explicitly cites and extends that canon: OWASP Top 10 for Agentic Applications, Google DeepMind's CaMeL, Meta's "Agents Rule of Two," Beurer-Kellner et al.'s six design patterns, and OpenAI's Instruction Hierarchy.

We deliver five things. First, a deep mapping of ten memory-corruption concepts to their agentic counterparts, with primary sources and analogy-strength assessments. Second, a parallel mapping of seven capability-security and social-engineering framings. Third, a full inventory of tools, testbeds, and frameworks for labs at every educational level from high school through graduate. Fourth, curriculum scaffolding and assessment guidance grounded in the pwn.college and picoCTF pedagogy literature. Fifth, the open problems where students can contribute and where the parallel misleads.

Two findings frame the contribution. First, every major classical security concept has a direct, teachable analogue in agentic AI, and most 2024-2026 defensive papers explicitly name their classical ancestor. The curriculum is architecturally well-grounded. Second, the analogy breaks in five specific places that students must learn explicitly: syntactic-vs-semantic validation, deterministic-vs-probabilistic execution, no hardware W^X analogue, persistence and self-reinforcement, and broken evaluation methodology. Nasr, Carlini, Tramèr et al. (*The Attacker Moves Second*, arXiv:2510.09023, October 2025) broke twelve recent model-level defenses at ≥90% attack success. No model-hardening defense has yet survived adaptive attack. The only durable defenses are system-level architectural ones: CaMeL, MELON, and plan-then-execute. Teaching the spine *and* the breakage is the curriculum's central pedagogical commitment.

Part 1, The architectural parallel as pedagogical spine

1.1 The foundational claim

Modern CPUs have implemented variants of Harvard-style instruction/data separation (W^X, NX bits, DEP, CFI, shadow stacks) precisely because the von Neumann architecture's unified memory makes classical exploitation possible. **Transformer LLMs have no analogous mechanism at the representation level**: every token. Whether it came from the system prompt, the developer's scaffold, the user's question, a retrieved document, or a tool's return value. Passes through the same self-attention layers and competes for behavioral control. The UK NCSC puts this as flatly as possible: "Current large language models simply do not enforce a security boundary between instructions and data inside a prompt." Simon Willison, who coined "prompt injection" in September 2022 as a deliberate echo of SQL injection, has since refined his position: **the analogy motivated the field but misrepresents the fix**, because "all of the previous injection attacks like SQL injection and command injection - we know how to fix them. With prompt injection it's fundamentally about how large language models function."

The pedagogical value is that the *pattern* (treating untrusted bytes/tokens as instructions) is common enough to run a full course on, while the *divergences* sharpen students' understanding of what makes classical defenses work and why agentic security is harder.

1.2 Ten classical defenses and their agentic counterparts

The following table is the spine of the curriculum. Each row is a module; each row also carries an honest assessment of how strong the analogy is, because half the pedagogical payload is in learning where analogies mislead.

#	Classical concept	Agentic counterpart	Key primary sources	Analogy strength
1	Buffer overflow / injection as instruction/data confusion	Direct and indirect prompt injection	Willison “Prompt injection” (simonwillison.net, Sep 2022); Greshake et al., “Not what you’ve signed up for,” arXiv:2302.12173 (AISec 2023)	**Strong at the conceptual level, misleading at the fix level**
2	Stack canaries, StackGuard	Spotlighting, StruQ delimiters, prompt canaries	Hines et al., “Spotlighting,” arXiv:2403.14720; Chen et al., “StruQ,” arXiv:2402.06363 (USENIX Security 2025); Rebuff canary-token design	Strong mechanically (both mark a trust boundary); weak cryptographically (LLM “boundaries” are learned soft constraints, not hardware-enforced)
3	DEP / W^X (Data Execution Prevention)	Control/data-plane separation: Dual-LLM, CaMeL, plan-then-execute	Willison, “Dual LLM pattern” (Apr 2023); Debenedetti et al., CaMeL, arXiv:2503.18813; Beurer-Kellner et al., arXiv:2506.08837	Conceptually tight; CaMeL is the closest thing to an architectural W^X for agents
4	ASLR (Address Space Layout Randomization)	Randomized smoothing, paraphrasing, SemanticSmooth	Robey et al., “SmoothLLM,” arXiv:2310.03684; Cao et al., RA-LLM, arXiv:2309.14348; Hu et al., “SemanticSmooth,” arXiv:2402.16192	Strong for *optimization-based* attacks (GCG); useless against semantic attacks (PAIR, persuasive roleplay)
5	Control Flow Integrity (CFI)	Plan/intent verification, intent capsules, Prompt Flow Integrity	Kim et al., “Prompt Flow Integrity,” arXiv:2503.15547; Wang et al., “AgentSpec,” arXiv:2503.18666; OWASP ASI 2026 “intent capsules”; Microsoft FIDES	**Explicit**. Papers literally use the term; tightest mapping in the set
6	Return-Oriented Programming (ROP)	Tool-chaining / “living off the tools” / MCP gadget composition	Invariant Labs “MCP Tool Poisoning” (Apr 2025); Rehberger’s Embrace The Red posts; MCPTox arXiv:2508.14925; MCPLib arXiv:2508.12538	**Very strong**. Reuses already-authorized building blocks; emergent behavior from legitimate parts
7	Format-string vulnerabilities	Prompt template injection, string composition	[illegible]	[illegible]

Table 1: Ten classical systems defenses and their agentic AI counterparts, with key primary sources and analogy-strength assessments per row.

1.3 Module-by-module depth

Buffer overflow → prompt injection. The anchor analogy. Simon Willison's September 2022 coinage is the canonical starting point; Greshake et al.'s "Not what you've signed up for" is the academic foundation. Greshake's observation. *"processing retrieved prompts can act as arbitrary code execution"*. Is the textual restatement of Morris-worm-era insight. The instructional payoff is showing students that the same system that makes LLMs useful (interpreting natural language as instruction) is the same system that makes them exploitable (interpreting *any* natural language as instruction).

Stack canaries → spotlighting / StruQ / canary tokens. Microsoft Research's *Spotlighting* paper (Hines et al., arXiv:2403.14720) introduces three tagging techniques. Delimiting (literal `<`, `>` markers), data-marking (per-token markers), and encoding (Base64-style encoding of data), that teach the model to recognize which parts of the context are "data" versus "instruction." StruQ (Chen et al., arXiv:2402.06363) adds reserved special tokens that are filtered from user data so an attacker cannot forge them, plus structured instruction tuning. Rebuff (Protect AI) uses *canary tokens*. Secret strings inserted into prompts that, if echoed, indicate a successful extraction attempt. **All three mechanisms are weak in the same way stack canaries were weak under info leaks**: adaptive attackers can learn the marker tokens, reconstruct them with optimization (AMS result), or encode around them.

DEP/W^X → control/data plane separation. The most important single module. Willison's April 2023 *Dual LLM Pattern* proposes that a **Privileged LLM** has tool access but can only see trusted input, while a **Quarantined LLM** handles untrusted input but has no tools. CaMeL (Debenedetti et al., arXiv:2503.18813, March 2025) is the *industrial realization*: a privileged LLM translates the user's request into a Python program executed by a **custom interpreter that tracks capability tags (provenance) on every value**; a quarantined LLM is invoked only to summarize untrusted content; a policy engine checks capabilities at every tool call. **CaMeL is literally CFI + IFC + capabilities applied to an LLM agent**, and its authors cite Abadi, Denning, and Anderson explicitly. The 77% provably-secure task-completion rate on AgentDojo (vs. 84% undefended) is the quantitative headline: you pay a ~7-point utility tax for a formal security guarantee.

ASLR → randomized smoothing. SmoothLLM (Robey et al., arXiv:2310.03684) randomly perturbs multiple copies of the input, runs each through the LLM, and majority-votes. It collapses GCG attack-success from ~100% to <1% on Vicuna. RA-LLM (Cao et al., arXiv:2309.14348) uses Monte-Carlo token-dropping. **These defenses are strong exactly where ASLR is strong**, against attackers who need determinism (GCG suffixes), and weak where ASLR is weak: semantic attacks (PAIR, persuasive roleplay, many-shot) survive perturbation because the *meaning* carries the payload, not the exact tokens. This parallels JIT-ROP bypassing ASLR via information leak.

CFI → plan/intent verification. The tightest mapping. Kim, Song et al.'s *Prompt Flow Integrity* (arXiv:2503.15547) deliberately names CFI. AgentSpec (Wang et al., arXiv:2503.18666) adds formal runtime verification to LangChain. OWASP ASI 2026 promotes **Intent Capsules**. Signed, immutable envelopes binding the agent's declared goal + constraints + context to each execution cycle, as "a mandatory architectural requirement." Microsoft FIDES enforces information-flow labels + confidentiality policies before

each tool call. The Beurer-Kellner et al. paper arXiv:2509.08646 explicitly frames plan-then-execute as "establishing control-flow integrity" for indirect prompt injection.

ROP → tool chaining / living off the tools. The second-tightest mapping. Johann Rehberger's series at embracethered.com is the practitioner canon: *"AI Kill Chain in Action: Devin AI Exposes Ports"* (Aug 2025); *"Cross-Agent Privilege Escalation: When Agents Free Each Other"* (2025); *"Agent Commander: Your Agent Works For Me Now"* (2026). Academic treatments include **MCPTox** (arXiv:2508.14925, 1,312 malicious cases across 11 categories), **MCPLib** (arXiv:2508.12538, 31 attacks in 4 classes including Multi-Tool Cooperation Attack), and *"When MCP Servers Attack"* (arXiv:2509.24272) defining control-flow hijack where a tool's return value steers the agent into additional tool calls. Invariant Labs' April 2025 **Tool Poisoning Attacks** disclosure is the foundational practitioner work, and the mcp-scan tool (now Snyk agent-scan) operationalizes defense. The ROP analogy is tight because **both reuse already-authorized executable material** (bytes marked X / tools granted to the agent) to achieve behavior never intended.

Shellcode → injected instructions. GCG (Zou et al., arXiv:2307.15043) is the canonical adversarial suffix paper. Essentially "compiled bytes for LLMs." The metaphor carries pedagogical weight but misleads if pressed: CPU shellcode is deterministic and Turing-complete; GCG suffixes are fragile, template-dependent, and often fail to transfer. OpenAI's December 2025 public acknowledgment that prompt injection against AI browsers "may never be fully solved" underscores that the *hardware* boundary between data and instructions, which shellcode era defenses eventually produced (NX, W^X). Has no clean LLM analogue.

Format-string → template injection. This is *not* analogy but literal reuse. LangChain's Jinja2 template handling produced CVE-2025-65106 (attribute-access template injection), CVE-2025-68664 ("LangGrinch" LangChain Core serialization, CVSS 9.3, ~847 million affected downloads), and CVE-2025-9556 (LangChainGo / Gonja SSTI, CVSS 9.8). **The Jinja `{{ ''.__class__.__bases__[0].__subclasses__()[147]...__globals__['popen']('dir').read() }}` payload works unchanged inside a LangChain prompt template file**, proving that the AI framing adds a *layer on top of* classical SSTI rather than replacing it.

Heap spraying → memory/context poisoning. PoisonedRAG (Zou et al., arXiv:2402.07867) shows that injecting a small number of crafted texts into a knowledge base suffices to force attacker-chosen answers. An optimization problem essentially identical to heap-spray probability. The Anthropic/UK AISI/Alan Turing Institute October 2025 study (arXiv:2510.07192) shows that **250 malicious documents suffice to backdoor LLMs from 600M to 13B parameters** regardless of training-set size. Poisoning cost is *constant*, not proportional. OWASP ASI06 elevates persistent cross-session memory poisoning to a first-class risk. **The critical divergence**: heap spray is session-local and transient; memory poisoning is *stateful and self-reinforcing*. Compromised agents "actively defend poisoned memories as legitimate when questioned" (Christian Schneider). The closer classical analogue is persistent rootkits.

NOP sleds → obfuscation / padding. Wei, Haghtalab & Steinhardt's *"Jailbroken"* paper (arXiv:2307.02483) documents Base64, ROT13, leetspeak, Morse, and payload division as "mismatched generalization" attacks. Pre-training taught decoding, safety training didn't. Anthropic's *Many-Shot Jailbreaking* (April 2024) uses hundreds of faux Q/A pairs to exhaust safety priors following a power law. Rehberger's ASCII Smuggler exploits

invisible Unicode Tag characters (U+E0000-U+E007F). Affected Microsoft 365 Copilot (patched July 2024) and Grok API (Dec 2024). *"Plentiful Jailbreaks with String Compositions"* (arXiv:2411.01084) unifies these as invertible transformations under combinatorial search. The NOP-sled analogy is weakest here. **obfuscation payloads are semantically active, not empty padding**, and polymorphic/metamorphic malware is often a closer classical reference.

1.4 What the SQL-injection analogy gets right and wrong

SQL injection was effectively solved by **parameterized queries**, the database engine enforces instruction/data separation at a level the application cannot bypass. The NCSC blog post *"Prompt injection is not SQL injection (it may be worse)"* states plainly: *"Whilst the comparison of prompt injection to SQL injection can be tempting, it's also dangerous. SQL injection can be properly mitigated with parameterised queries, but there's a good chance prompt injection will never be properly mitigated in the same way. The best we can hope for is reducing the likelihood or impact of attacks."* Willison himself now says: *"If I use parameterized SQL queries my systems are 100% protected… If our measures against SQL injection were only 99% effective none of our digital activities involving relational databases would be safe."*

The engineering conclusion, and the defining pedagogical point. Is that **natural language has no parameterization primitive**. Every defense at the model layer is probabilistic; the only way to get the SQL-injection feeling back is to impose *structural* separation at the agent architecture layer (CaMeL, Dual-LLM, plan-then-execute).

Part 2, Capability security and social engineering

2.1 The confused deputy problem

Norm Hardy's 1988 paper *"The Confused Deputy (or why capabilities might have been invented)"* (ACM SIGOPS OSR 22:4, https://dl.acm.org/doi/10.1145/54289.871709) tells the FORTRAN-compiler-at-Tymshare story: a compiler holds authority both from the invoker (who can name an output file) and from the system (to write a billing file), and a malicious invocation makes it overwrite billing records. Hardy's diagnosis. *"The fundamental problem is that the compiler runs with authority stemming from two sources... It has no way to keep them apart"*. Is word-for-word a description of agentic AI systems. The agent executes with the user's OAuth tokens, session cookies, filesystem credentials, and tool scopes, but takes its *instructions* from any content it processes, including attacker-controlled retrieved documents, emails, or tool returns.

The mapping is explicit throughout the literature. Willison's *lethal trifecta* formulation. Private data + untrusted content + external communication. Is the confused-deputy problem restated; an agent becomes exploitable at the exact moment all three are present. Modern examples: **EchoLeak** (CVE-2025-32711, Microsoft Copilot zero-click exfiltration via email); **ChatGPT Operator exfiltration via Hacker News** (Rehberger, Feb 2025); the **Supabase MCP** incident (July 2025) where prompt injection in a ticket gave SQL access; **Devin AI** port-exposure (Rehberger, Aug 2025). Formal treatments, Rajani et al. "On access control, capabilities…" (CSF 2016, https://people.mpi-sws.org/~dg/papers/csf16-caps.pdf). Define *CDA-freedom* and prove capability systems prevent confused-deputy attacks.

For the curriculum, Hardy 1988 is *the* seed reading. It is short, concrete, and 37 years ahead of its time.

2.2 Object capabilities and POLA

Mark Miller's 2006 Johns Hopkins dissertation *Robust Composition* (http://erights.org/talks/thesis/markm-thesis.pdf) is the canonical text on object-capability security, building on Dennis & Van Horn (1966). The core insight. Authority transfers only by explicit reference-passing. Is exactly what an ideal agentic tool-access model should enforce. Miller's *"Capability Myths Demolished"* dismantles misconceptions (e.g., that capabilities can't limit propagation).

CaMeL is the headline contemporary application. The paper's own text: *"CaMeL associates, to every value, some metadata (commonly called capabilities in the software security literature) to restrict data and control flows, giving the possibility to express what can and cannot be done with each individual value by using fine-grained security policies."* CaMeL is hybrid ocap + IFC. Closer to labels with a capability-based access-policy layer

at tool boundaries, and the cap-talk mailing-list discussion (https://groups.google.com/g/cap-talk/c/YmZj1EhY4Uk) was frank about whether the "capabilities" are genuine ocaps. Either way, the design is a descendant of Miller's work and cites it directly.

Other concrete applications: **Tenuo** (cryptographic warrants with offline attenuation for LLM tool calls, listed on github.com/dckc/awesome-ocap); MCP manifests with signed tool permissions; per-call revocable tokens via membrane patterns.

2.3 Principle of least privilege → Least Agency

Saltzer & Schroeder 1975, *"The Protection of Information in Computer Systems,"* (https://www.cs.virginia.edu/~evans/cs551/saltzer/) defines least privilege: *"Every program and every user of the system should operate using the least set of privileges necessary to complete the job."* The other seven principles. Economy of mechanism, fail-safe defaults, complete mediation, open design, separation of privilege, least common mechanism, psychological acceptability. Are the charter of systems security.

OWASP Top 10 for Agentic Applications 2026 (released December 9, 2025) extends this as **Least-Agency**: *"An extension of the Principle of Least Privilege. Avoid unnecessary autonomy. Agents should only be granted the minimum level of autonomy required to complete their defined task."* Paired with **Strong Observability** ("comprehensive visibility into what agents are doing, why, and which tools they are invoking") and **Intent Capsules**, this constitutes the architectural principle set of the agentic security era. Practical operationalizations include Just-in-Time Agency (autonomy granted only for the specific short-lived task) and an "agency budget" concept.

2.4 Taint tracking and information flow control

The classical lineage runs from Perl's `-T` taint mode and Ruby's (now-removed) `$SAFE` levels through Denning & Denning's lattice model of secure information flow (1976-1977, https://dl.acm.org/doi/10.1145/360051.360056), Myers' JFlow / Jif (POPL 1999), and the Sabelfeld & Myers 2003 JSAC survey *"Language-based Information-Flow Security."* **CaMeL is this lineage applied to LLM agents**: its Python interpreter builds a data-flow graph, labels each value with capabilities (origin, readers, allowed sinks), and at every tool call checks that argument capabilities satisfy the tool's policy. If a tool requires a `trusted` email recipient and the value has only `untrusted`, the call is blocked.

2.5 Sandboxing

The classical lineage. Chroot (1979) → jails (1999) → namespaces/cgroups (2008) → seccomp → Docker (2013) → gVisor (2018) → Firecracker (2018). Has a direct analogue in the 2024-2026 agent-sandbox landscape: **E2B** (Firecracker microVMs, ~125-150ms cold start), **Modal** (gVisor with GPU), **Daytona** (Docker-based, sub-90ms cold start), **Cloudflare Sandbox** (V8 + Durable Objects), **Vercel Sandbox** (Firecracker). Pydantic's **mcp-run-python** project. A Deno-wrapped Pyodide sandbox. Was *retired and archived* in late 2025 with an unusually candid statement from the maintainers: *"there's just no safe*

way to run Python within pyodide safely with reasonable latency… Python code running in pyodide can run arbitrary javascript." The successor project is **Monty**, a minimal Rust-written Python interpreter. This retirement is itself a teachable moment: isolation that is "good enough" for classical untrusted code is insufficient for agent code paths that include LLM-generated payloads.

2.6 Why SQL injection is not the right analogy (again)

Covered in §1.4 above. The summary is: SQL injection was solvable because the database engine provides parameterization as a primitive; natural language has no such primitive. The NCSC and Cisco both state this unambiguously, and Willison himself now says the analogy *"is extremely difficult, if not impossible, to implement on the current architecture of large language models."* The architectural defenses (CaMeL et al.) do restore something like parameterization, but at the *agent* layer, not the model layer.

2.7 Social engineering parallels

Two directions matter. **First, humans attacking agents** via persuasion. Zeng, Lin, Zhang et al.'s *"How Johnny Can Persuade LLMs to Jailbreak Them"* (ACL 2024, arXiv:2401.06373) catalogs 40 persuasion techniques from Cialdini-style psychology and communication theory; their Persuasive Adversarial Prompts achieve >92% attack success on GPT-4, GPT-3.5, and Llama-2-7b-Chat, with stronger models sometimes more vulnerable than weaker ones. DAN, Grandma Exploit, Evil Confidant, and the broader *persona-modulation* family (Shah et al., arXiv:2311.03348) operationalize this at scale. One paper breaks 42 of 43 restricted categories for under $3 per model. CyberArk's *"Operation Grandma"* used the pattern against the Israeli Ministry of Labor's chatbot to generate ransomware.

Second, agents attacking humans, OWASP's **ASI09 Human-Agent Trust Exploitation**. The OWASP 2026 text: *"Intelligent agents can establish strong trust with human users through their natural language fluency, emotional intelligence, and perceived expertise, known as anthropomorphism. Adversaries or misaligned designs may exploit this trust… By leveraging authority bias and persuasive explainability, attackers can bypass oversight… The agent acts as an untraceable 'bad influence,' manipulating the human into performing the final, audited action."* The practical recommendation (Auth0): **never let the agent's own chat surface be where permission is granted**. Route consent through a separate authenticated flow. This is the agent equivalent of "never trust user input for authorization decisions."

Part 3, Hands-on tools and lab environments

The tooling landscape for lab instruction is unusually mature for a field this young, and it divides naturally into six categories plus the classical-exploitation platforms that form the systems-security prerequisite.

3.1 Prompt-injection playgrounds (approachable from high school up)

Gandalf (Lakera, gandalf.lakera.ai) is the genre-defining entry point: seven base levels plus "Gandalf the White," "Agent Breaker" (tool-call attacks), and adventure tracks. Free, browser-based, ~9M interactions in its first 20 days. Levels 1-3 are suitable for high-school classrooms; level 7-8 and Agent Breaker challenge graduate students. **HackAPrompt 2.0** (hackaprompt.com; dataset MIT-licensed on HuggingFace; Schulhoff et al. EMNLP 2023 Best Theme Paper) is the tiered competitive platform, now with agentic/tool-use tracks and a $100k+ prize pool; the original contest collected 600K+ adversarial prompts from 2,800+ participants and produced the **29-technique taxonomy** that functions as the curriculum's "vulnerability ladder" for prompt injection. **Tensor Trust** (Toyer et al., ICLR 2024; tensortrust.ai) is the defense-*and*-attack construction environment. Students write defense prompts and attack others' defenses, producing the largest public human-authored adversarial dataset (563K attacks, 118K defenses). **Prompt Airlines** (Wiz Research) is the best five-challenge bridge to agentic scenarios (tool-call exfiltration, multimodal injection). **Microsoft AI Red Teaming Playground Labs** (github.com/microsoft/AI-Red-Teaming-Playground-Labs, MIT) ships twelve hands-on labs with PyRIT integration and Jupyter notebooks. It is the single best open-source undergraduate lab kit. **Doublespeak** (Forces Unseen) pairs with the open-source *LLM Hackers Handbook* for structured self-study. **Gray Swan Arena** offers competitive red-teaming with real prize pools and has partnered with US AISI, UK AISI, DeepMind, and Anthropic.

3.2 Agentic-security testbeds (undergraduate-to-graduate)

AgentDojo (Debenedetti et al., NeurIPS 2024 Datasets & Benchmarks; ETH SPY Lab + Invariant Labs; agentdojo.spylab.ai) is the de-facto benchmark: 97 realistic tasks × 629 security test cases across four environment suites (email, banking, travel, Slack) with 74 tools. It was adopted by the US and UK AI Safety Institutes for Claude 3.5 Sonnet red-teaming. **InjecAgent** (Zhan et al., ACL 2024 Findings) provides 1,054 test cases across 17 user tools and 62 attacker tools. **Agent Security Bench (ASB)** (Zhang et al., ICLR 2025) is the broadest, 10 scenarios, 400+ tools, 27 attack/defense methods, 13 LLM backbones, and introduces the Net Resilient Performance metric balancing utility and security. **AgentHarm** (Andriushchenko et al., Gray Swan + UK AISI, ICLR 2025) covers 110 harmful tasks expanded to 440 across 11 harm categories; MIT-licensed with a research-restricted clause. **BIPIA** (Microsoft, KDD 2025) is the indirect-prompt-injection benchmark. For MCP security specifically, **mcp-scan** (Invariant Labs, acquired by Snyk; github.com/invariant-

labs-ai/mcp-scan) is the go-to scanner for tool poisoning, shadowing, rug-pulls, toxic flows, and cross-origin escalation; it auto-discovers Claude Desktop, Cursor, Windsurf, Gemini CLI, and VS Code configurations.

3.3 Red-teaming frameworks

Microsoft PyRIT (github.com/microsoft/PyRIT, MIT) is the production-grade reference with orchestrators, converters, scorers, memory, multi-turn strategies (Crescendo, PAIR, Tree-of-Attacks) and cross-provider support. **NVIDIA garak** (Apache 2.0) is "nmap for LLMs". Probe-and-detector model, widely deployed. **Promptfoo** (MIT) is the most instructor-friendly because of its web UI and plugin system mapping to OWASP LLM Top 10 and MITRE ATLAS. **Giskard** (Apache 2.0) has first-class agent support in v3 and is used in DeepLearning.AI's "Red Teaming LLM Applications" course. **Meta Purple Llama / CyberSecEval 4** is the most comprehensive benchmark suite, adding CyberSOCEval (malware analysis + threat-intel reasoning with CrowdStrike) and AutoPatchBench.

3.4 Classical exploitation platforms (the systems-security backbone)

pwn.college (Yan Shoshitaishvili & Connor Nelson, ASU; CC-BY-NC content) powers ASU's CSE 365/466 and is the leading belt-structured binary-exploitation curriculum, 1000+ challenges, 145 countries, a $4.5M DARPA grant in 2025. **picoCTF** (CMU CyLab + PPP, 2013-present) is the world's largest student CTF; its GenCyber teacher-training program makes it the canonical high-school pipeline. **Exploit Education Phoenix** (supersedes Protostar) is the textbook-style stack-overflow-through-heap lab VM. **ROP Emporium** isolates return-oriented programming across eight progressive challenges on four architectures. **pwnable.kr and pwnable.tw** are the classic Korean and Taiwanese wargames. **OverTheWire Bandit/Natas** are ideal high-school entry; Narnia/Behemoth/Utumno carry students up the binary-exploitation ladder. **Microcorruption** (embedded MSP430 exploitation) is notably HS-accessible. **CSAW CTF** (NYU OSIRIS Lab) is the largest student-run CTF. **HackTheBox** (commercial) covers the full pentest kill-chain.

3.5 Agent sandboxes

E2B (Firecracker microVM, open-source core), Modal (gVisor + GPU), Daytona (Docker + sub-90ms cold start; $24M Series A Feb 2026), Cloudflare Sandbox SDK (edge Durable Objects), Vercel Sandbox (Firecracker, 45-min cap). Avoid mcp-run-python for any lab deployment (see §2.5). For DIY self-hosting, Docker + gVisor is the baseline, Kata Containers or Firecracker the microVM upgrade. A practical heuristic (cited by Northflank and Stealth Cloud): *"Default to microVMs for untrusted code. Relax to gVisor or containers only when threat model justifies it."*

3.6 Defensive frameworks

NVIDIA NeMo Guardrails (Apache 2.0) is the reference open-source guardrail stack with input/dialog/retrieval/execution/output rails, Colang DSL, and integrated garak scanning. **Meta Llama Guard 4** (12B multimodal) and **Prompt Guard 2** (86M / 22M) are the best open classifier models. **Protect AI LLM Guard** (MIT) provides 15 input + 20 output scanners and is the industry-standard OSS guardrail library. **Rebuff** (Apache 2.0, labeled "prototype") is pedagogically valuable for teaching layered-defense architecture. Heuristics + LLM detection + VectorDB + canary tokens. **Guardrails AI** (Apache 2.0) ships a Hub of pluggable validators and benchmarks 24 guardrails in the Guardrails Index. **Microsoft Presidio** is the PII-redaction reference. **Lakera Guard** and **Google Model Armor** are the production commercial baselines to compare student-built defenses against.

3.7 Benchmarks and datasets

AdvBench (Zou et al.) is the GCG seed set. **HarmBench** (Center for AI Safety, ICML 2024, MIT) standardizes red-teaming against 510 harmful behaviors. **JailbreakBench** (Chao et al., NeurIPS 2024, MIT) maintains a live leaderboard of 100 representative misuse behaviors. **Do-Not-Answer** (LibrAI/UMelb, EACL 2024) is the first open evaluation set for LLM refusal. **PINT** (Lakera, Apache 2.0 code with private prompts) is the neutral detection benchmark. **TrustLLM** covers eight trustworthiness dimensions across 30+ datasets.

3.8 Tool × education-level summary

Category	High school	Undergraduate	Graduate
Prompt-injection playground	Gandalf levels 1-3, Prompt Airlines, Doublespeak	HackAPrompt 2.0 full path, Tensor Trust, MS AI Red Teaming Playground Labs	Gray Swan Arena, HackAPrompt agentic tracks
Agentic testbed	-	InjecAgent (quick-start), BIPIA	AgentDojo, ASB, AgentHarm
Red-teaming framework	, (instructor-led demo)	Promptfoo, garak	PyRIT, CyberSecEval 4
Classical exploitation	picoCTF, OverTheWire Bandit/Natas, Microcorruption	pwn.college white→yellow belt, Phoenix, CMU Attack Lab	pwn.college brown→black, ROP Emporium, pwnable.tw
Agent sandbox	Cloudflare Sandbox (guided)	E2B, Daytona	E2B BYOC, Modal, DIY Firecracker
Defensive framework	Presidio (applied)	LLM Guard, Guardrails AI	NeMo Guardrails + custom research
Benchmark	Do-Not-Answer (reading)	JailbreakBench (reading)	HarmBench, TrustLLM, JBB leaderboard submissions

Table 2: Hands-on tools mapped to education level. Seven tool categories scaffolded across high-school, undergraduate, and graduate progression.

Part 4, Curriculum design and pedagogy

4.1 Existing university courses

The peer-reviewed AI-security course landscape is still sparse as of April 2026, but a workable composite is available. **MIT 6.858 / 6.5660** ("Computer Systems Security") remains the systems-security backbone: Lab 1 in week 1 is a buffer overflow, followed by defenses, privilege separation, OS security, and web security. Note that MIT 6.S983 is *hardware* security despite surface similarity to "AI security". Do not conflate. **Stanford CS155** is the twin undergraduate course; **CS329T** ("Trustworthy Machine Learning," Mitchell/Datta/Taly since 2023) covers groundedness, calibration, explainability, privacy, fairness, and adversarial attacks on LLMs. **UC Berkeley CS 294-131** ("Trustworthy Deep Learning," Dawn Song) has rotated through security, privacy, robustness, and fairness; Berkeley CLTC publishes accessible adversarial-ML explainers. **ETH SPY Lab** (Florian Tramèr) teaches PETs and produced AgentDojo; Tramèr's "Negative Progress in Machine Learning Security" lecture is an excellent seed for graduate seminars. **CMU** leads K-12 scaffolding (picoCTF/GenCyber) and systems-security labs (CS:APP Bomb Lab, Attack Lab). **ASU** (Shoshitaishvili) operates pwn.college. **Nicholas Carlini's** adversarial ML reading lists (at nicholas.carlini.com) are the canonical entry points, "just-the-basics" (5 papers), "quick-introduction" (~10), and a live list of 1000+ adversarial-example papers.

4.2 Scaffolding from high school through graduate

The picoCTF experience. Founded 2013 with ~10,000 students in its first event, now approaching 1M cumulative learners. Establishes three central claims for scaffolding: (1) gamified CTFs with progressive difficulty inside categories produce flow states that drive voluntary investment; (2) offense-first framing accelerates systems-thinking; (3) NSA GenCyber teacher training is the key lever for classroom adoption. For the graduate-level step, Nelson & Shoshitaishvili's SIGCSE 2024 paper *"PWN the Learning Curve"* crystallizes the pwn.college pedagogy: **integration modules** chain radically different vulnerability classes (race condition → heap UAF → ROP → shellcode) to mirror real exploit chains; concepts must be *integrated* but need not be *taught* concurrently; and **self-guiding challenge variants** that print internal state before requiring students to drive gdb themselves remove a massive novice barrier. The core CTF-pedagogy research (Vykopal et al., SIGCSE 2020; Weiss et al., SIGCSE 2016; O'Leary, SIGCSE 2017) flags three failure modes to avoid. Overly difficult challenges, ambiguous challenges, limited progress feedback.

4.3 Lab design principles

The distilled principles from this literature are: **progressive difficulty with short feedback loops** (every challenge one conceptual step beyond the previous); **decouple**

concept-learning from tool-learning (scaffold gdb/strace fluency separately); **isolation** (per-user Docker dojos, per-student binaries); **unique per-student artifacts** to prevent answer sharing (Bomb Lab's key trick); **instant authoritative scoring**; **partial-credit scaffolding** for novices; and a **capstone that is open-ended**. These transfer cleanly to prompt-injection labs: Gandalf's level-by-level progression, AgentDojo's (user task × injection × attack × defense) matrix, and HackAPrompt 2.0's taxonomic tracks are the direct analogues of the pwn.college belt system.

4.4 Assessment

Flag-submission scoring is formative and summative in one. pwn.college's "A+ if you solve all challenges" removes grade anxiety while requiring real investment. Authentic assessment (Weiss et al.) adds reflection portfolios, what did the student try, why did it fail, what did they learn. For prompt injection, **ASR-on-benchmark** plus a **defense-evaluation report using adaptive attacks** is the right graduate-level rubric, directly inheriting from Carlini's *"On Evaluating Adversarial Robustness"* (arXiv:1902.06705). A rubric for adversarial thinking, grounded in Ceraolo et al.'s 2023 *Journal of Cybersecurity* study, assesses monitoring (noticing anomalies), investigating (tracing to a concrete bug), and evaluating (situating in threat model). A **responsible-disclosure deliverable**. A student-authored disclosure to a real vendor's bug bounty. Validates transfer at the graduate level.

4.5 Safety for teaching offensive techniques

Four pillars. **Ethics frameworks**: ACM Code of Ethics (especially 1.2, 2.8, 2.9), IEEE-CS/ACM Software Engineering Code, EC-Council CEH Code. **picoCTF safety practices for minors**: all infrastructure CMU-hosted and reset regularly; no student instructed to attack external services; GenCyber teacher training includes an ethics/responsible-disclosure module; classroom norms forbid using learned techniques outside the platform. **Responsible red-teaming**: Anthropic's *Frontier Threats Red Teaming* program and *Responsible Scaling Policy* are the most explicit external frameworks (work with domain experts, test on secure non-user-facing interfaces, coordinated disclosure over public disclosure); OpenAI's *Approach to External Red Teaming* defines red-teaming as "structured testing… to find flaws and vulnerabilities… using adversarial methods." **Dual-use publishing norms**: publish attacks with reproducibility but not one-click weaponization; coordinate disclosure; release datasets under responsible-use terms.

4.6 Recommended age-appropriate progression

High school (grades 9-12). Systems thinking and the security mindset. Memory track: no real exploitation; picoCTF reverse-engineering and basic-binary categories; defuse a 3-phase binary-bomb clone. Prompt-injection track: Gandalf 1-8 with a reflection identifying HackAPrompt taxonomic labels for each successful attack; a short Schneier + Anthropic responsible-disclosure reading; Prompt Airlines for an agent-adjacent scenario. Ethics module with parental consent if attacks go beyond the sandbox.

Undergraduate (sophomore-senior). *Intro course:* CMU Bomb Lab + Attack Lab; Stanford CS155 Lab 1 or MIT 6.5660 Lab 1; pwn.college white/yellow belt. Parallel prompt-injection track: HackAPrompt 2.0 full path; build a naive RAG chatbot, inject it directly and indirectly, add sandwich prompting and measure ASR change; Tensor Trust for defense construction. *Advanced UG:* pwn.college brown belt (heap, kernel, sandbox escape); Shacham's *"Geometry of Innocent Flesh on the Bone"* and Carlini-Wagner *"ROP is Still Dangerous."* Parallel agentic track: full AgentDojo assignment reporting utility-under-attack vs. ASR Pareto curves; MCP-scan audit of a student-built MCP server; read AgentDojo, InjecAgent, Greshake, and Carlini's *"Are Aligned Neural Networks Adversarially Aligned?"*; a 5-page defense-evaluation report with adaptive-attack methodology. A 2-3 week ethics unit covers ACM Code, CFAA/DMCA, coordinated disclosure, and AI-specific responsible-red-teaming per Anthropic/OpenAI guides. This maps to CAE-CD and begins satisfying CAE-CO Knowledge Units.

Graduate. Novel defense design under adaptive attack; formal methods; independent vulnerability discovery; responsible disclosure to vendor bug bounties. Memory track: Abadi CFI formalization; Carlini-Wagner *"Control-Flow Bending"* (USENIX 2015); Shoshitaishvili et al. *"angr"* (IEEE S&P 2016); syzkaller-driven kernel fuzzing. Parallel agentic track: reimplement CaMeL or a Beurer-Kellner design pattern; construct a novel Byzantine-MAS attack; submit to AgentDojo, AgentHarm, JailbreakBench leaderboards; conference-style paper targeting USENIX Security, IEEE S&P, CCS, NeurIPS D&B, or ACL. IRB training; frontier-threat red-teaming protocols; dual-use research statement in every paper. Maps to CAE-R and NICE Work Roles in Exploitation Analysis and Vulnerability Assessment.

Part 5, Open problems and where the analogy breaks

5.1 Where the memory-corruption analogy breaks

Five specific failure modes, each worth teaching explicitly because each corresponds to a defense that *won't* transfer:

1. **Syntactic vs. semantic validation.** Classical injection is a parsing ambiguity in a formal grammar. Fixable by escaping, parameterization, or type systems. Prompt injection is a property of *meaning* assigned by a probabilistic model; natural language has no formal grammar separating instruction from data. This is why "escape the user input" doesn't work for prompts.
2. **Deterministic vs. probabilistic execution.** The CPU executes a given byte sequence the same way every time. The LLM does not: GCG suffixes are fragile, template-dependent, and often fail to transfer. Defenses that depend on deterministic adversary behavior (e.g., signature-based) lose ground to combinatorial obfuscation (arXiv:2411.01084).
3. **No hardware W^X analogue.** Transformer self-attention mixes all tokens through the same weights. Architectural approximations (StruQ reserved tokens, Meta-SecAlign's new input role, OpenAI's instruction hierarchy) are *learned soft constraints*, not hardware-enforced. The October 2025 *"The Attacker Moves Second"* paper (Nasr, Carlini, Tramèr et al., arXiv:2510.09023) broke twelve such defenses at ≥90% ASR using adaptive gradient, RL, search, and human attacks.
4. **Persistence and self-reinforcement.** Heap corruption is transient. Reboot clears it. Memory poisoning (ASI06) is cross-session and self-defending: compromised agents actively defend poisoned memories when questioned (Christian Schneider). The closer classical analogue is rootkits or BIOS implants, not heap spraying.
5. **Broken evaluation methodology.** Classical security has cryptographic reductions and Kerckhoffs-principle evaluation. LLM security has neither, and the field is still learning what the AMS paper implies: *all static-attack-set evaluations are upper bounds on defense strength, not characterizations of it.*

5.2 Is prompt injection fundamentally unsolvable?

The consensus forming in 2025-2026 is **"at the model layer, yes; at the system layer, possibly not."** The pessimists include Greshake et al. ("difficult to achieve"), NCSC ("may simply be an inherent issue"), Meta AI (*"a fundamental, unsolved weakness in all LLMs"*), and Beurer-Kellner et al. (*"we believe it is unlikely that general-purpose agents can provide meaningful and reliable safety guarantees"*). The architectural optimists, CaMeL (arXiv:2503.18813), MELON (Zhu et al., ICML 2025), Balunović et al. (ICML 2024 NGAIS workshop). Show that restricted task classes can be provably defended via OS-

style containment, and Meta's *Agents Rule of Two* and Willison's *lethal trifecta* heuristic provide deterministic policy guidance: an agent may have at most two of {untrusted input, sensitive data, external communication}. Simon Willison's review of six design patterns (simonwillison.net/2025/Jun/13/) is the best practitioner framing. The engineering bet is that **abandoning model-hardening in favor of classical-security architectures** is the path that works.

5.3 Recent 2024-2026 defenses and their evidence base

The fullest accounting is the table below (drawn directly from the Part-5 subagent's research). The pattern is unmistakable: **every training-based defense falls to adaptive attack; only system-level architectural defenses survive.**

Defense	Year	Approach	Broken by adaptive attack (AMS, arXiv:2510.09023)?
StruQ (Chen et al., arXiv:2402.06363)	2024	Reserved tokens + structured instruction tuning	Yes
Instruction Hierarchy (Wallace et al., arXiv:2404.13208)	2024	4-level privilege-ranked SFT	Yes; see Wu et al., arXiv:2502.15851
SecAlign (Chen et al., arXiv:2410.05451, CCS 2025)	2024	DPO on (injection, secure, insecure) triples	Yes
Constitutional Classifiers (Anthropic, arXiv:2501.18837)	2025	Input/output classifiers from synthetic constitutional data	Partially; Constitutional Classifiers++ (arXiv:2601.04603) adds probe + ensemble
Meta-SecAlign (arXiv:2507.02735)	2025	New input role separating trusted user from untrusted external data	Yes
Prompt Guard 2 (Meta, 86M/22M)	2025	DeBERTa-v2 classifier	Yes (classifiers are inherently fragile)
Google Model Armor	2025	Vertex AI guardrail (PI detect, SDP, URL, PDF)	Yes (classifier-based)
Spotlighting (Hines et al., arXiv:2403.14720)	2024	Delimiting, data-marking, encoding	Yes
CaMeL (Debenedetti et al., arXiv:2503.18813)	2025	**System-level: privileged + quarantined LLM + capability interpreter**	**No. Structural guarantee**
MELON (Zhu et al., arXiv:2502.05174, ICML 2025)	2025	**Provable IPI defense via masking/planning**	**No. Architecture-level**
Beurer-Kellner design patterns (arXiv:2506.08837)	2025	**Action-Selector, Plan-Then-Execute, Map-Reduce, Dual-LLM, Code-Then-Execute, Context-Minimization**	**Pattern-dependent; architectural**
Meta Agents Rule of Two	2025	**Deterministic policy: at most 2 of {untrusted, sensitive, external}**	**Policy-level, not defeatable by single-agent attack**

Table 3: 2024-2026 defenses against prompt injection. Approach and survival under adaptive attack (AMS, arXiv:2510.09023). Training-based defenses fall under adaptive attack; system-level architectural defenses (CaMeL, MELON, Beurer-Kellner design patterns, Meta Agents Rule of Two) survive structurally.

5.4 The OpenAI instruction hierarchy in depth

Wallace, Xiao, Leike et al.'s *"The Instruction Hierarchy"* (arXiv:2404.13208, April 2024) is a landmark paper because it reframed prompt injection as a **privilege-inversion problem**: the system prompt, the user prompt, the tool output, and the model's own output should be treated as privilege tiers, with higher-privileged instructions winning conflicts. OpenAI trains this hierarchy via synthetic aligned/misaligned examples generated by context distillation (for aligned) and red-team LLMs + gradient attacks (for misaligned), then SFT + RLHF. Deployed in GPT-4o-mini (July 2024), then GPT-4o, o1, and GPT-5 (confirmed in the GPT-5-Codex system card, which cites the paper). OpenAI's Model Spec 2025 formalizes the hierarchy normatively. The critique comes from Wu et al.'s *"Failure of Instruction Hierarchies"* (arXiv:2502.15851), which shows that even hierarchy-trained models follow lower-priority instructions a significant fraction of the time when constraints are mutually exclusive but individually benign, and that AMS bypasses succeed at ≥90% ASR. **The hierarchy is the best model-layer defense to date, and it is still not sufficient.**

5.5 OWASP Top 10 for Agentic Applications 2026

Released December 9, 2025 by the OWASP GenAI Security Project – Agentic Security Initiative (Sotiropoulos, Katz, Del Rosario, Habler et al., genai.owasp.org). The ten items, ASI01 Agent Goal Hijack, ASI02 Tool Misuse & Exploitation, ASI03 Identity & Privilege Abuse, ASI04 Agentic Supply Chain Vulnerabilities, ASI05 Unexpected Code Execution, ASI06 Memory & Context Poisoning, ASI07 Insecure Inter-Agent Communication, ASI08 Cascading Failures, ASI09 Human-Agent Trust Exploitation, ASI10 Rogue Agents. Are documented with concrete incidents: **EchoLeak** (CVE-2025-32711, zero-click Copilot exfiltration, ASI01), **Amazon Q** destructive commands (ASI02), the **GitHub MCP exploit** (Invariant Labs 2025, ASI04), **CVE-2025-6514** OS command injection in mcp-remote (ASI05), the **Gemini Memory Attack** (Rehberger, ASI06), **Replit meltdown** where an agent deleted a production database (ASI10). The framework's two architectural principles. **Least-Agency** and **Strong Observability**, and its signature pattern, **Intent Capsules** (signed immutable envelopes binding goal, constraints, and context to each execution cycle), constitute the architectural spine any 2026-era curriculum should reflect.

5.6 Research frontier

The frontier problems suitable for student contribution are: (1) **provable prompt-injection defense for general-purpose agents**, CaMeL/MELON cover restricted classes only; (2) **adaptive-attack evaluation standards** for LLMs analogous to Carlini-Wagner 2017 for vision; (3) **Byzantine multi-agent tolerance under semantic faults**. See Schroeder de Witt et al.'s *"Open Challenges in Multi-Agent Security"* (arXiv:2505.02077), BlindGuard (arXiv:2508.08127), BlockAgents (ACM TCCE 2024), and the striking *"Dark Side of LLMs"* result (arXiv:2507.06850) that 82.4% of LLMs yield to injections from *peer agents* even when only 41.2% yield to direct injection; (4) **MCP protocol hardening**, Maloyan & Namiot (arXiv:2601.17549) flag missing capability attestation, unauthenticated sampling, and implicit trust propagation as protocol-level gaps; MCPSecBench and MCP-38 taxonomies cover the attack space; (5) **supply-chain defense at constant attack cost**, the Anthropic/UK AISI/Alan Turing Institute October 2025 study (arXiv:2510.07192)

shows 250 malicious documents suffice to backdoor LLMs regardless of scale; (6) **rogue-agent detection** (ASI10) where normal behavior itself varies nondeterministically; (7) **memory-poisoning persistence** and clear-on-taint semantics for long-running agents; (8) **human-agent trust calibration** (ASI09) as a behavioral-science question; (9) **economic/financial-DoS circuit breakers** for agents with asymmetric per-call cost.

Annotated bibliography of the most important sources

Foundational primary sources (classical). Saltzer & Schroeder (1975) “Protection of Information in Computer Systems”, the eight principles, especially least privilege; Hardy (1988) “Confused Deputy”. Three pages that predict agentic AI security; Denning (1976) “A Lattice Model of Secure Information Flow”, the IFC foundation CaMeL rests on; Miller (2006) *Robust Composition*, the object-capability canon; Abadi et al. (2005) on CFI, which intent-capsule papers cite directly.

Foundational primary sources (agentic). Willison’s September 2022 post coining “prompt injection” (simonwillison.net/2022/Sep/12/prompt-injection/); his 2023 “Dual LLM pattern” (simonwillison.net/2023/Apr/25/dual-llm-pattern/); his June 2025 “Lethal trifecta” (simonwillison.net/2025/Jun/16/the-lethal-trifecta/); his April 2025 “MCP has prompt injection security problems”, these four posts are the practitioner’s backbone. **Greshake et al.** “Not what you’ve signed up for” (arXiv:2302.12173, AISec 2023) is the academic foundation. **Zou et al.** “Universal and Transferable Adversarial Attacks” (arXiv:2307.15043) is the adversarial-suffix canon. **Wei, Haghtalab, Steinhardt** “Jailbroken” (arXiv:2307.02483) is the jailbreak-taxonomy canon.

2024-2026 defense papers (must-read). OpenAI Instruction Hierarchy (Wallace et al., arXiv:2404.13208); StruQ (Chen et al., arXiv:2402.06363, USENIX 2025); SecAlign (Chen et al., arXiv:2410.05451, CCS 2025); **CaMeL** (Debenedetti et al., arXiv:2503.18813, Google DeepMind, March 2025), the single most important paper to read for the architectural turn; AgentDojo (Debenedetti et al., arXiv:2406.13352, NeurIPS 2024 D&B); Beurer-Kellner et al. “Design Patterns for Securing LLM Agents” (arXiv:2506.08837, June 2025); Anthropic Constitutional Classifiers (arXiv:2501.18837); Microsoft Spotlighting (Hines et al., arXiv:2403.14720).

2024-2026 critical evaluation. Nasr, Carlini, Tramèr et al. “The Attacker Moves Second” (arXiv:2510.09023, October 2025), the single most important paper on evaluation methodology; Wu et al. “Failure of Instruction Hierarchies” (arXiv:2502.15851); Carlini’s talks and reading lists at nicholas.carlini.com.

Standards and governance. NIST AI 100-2 E2025 (Vassilev et al., March 2025); Alan Turing Institute CETaS “Indirect Prompt Injection” (Ruck & Sutton, Nov 2024); UK NCSC “Prompt injection is not SQL injection” (ncsc.gov.uk); OWASP Top 10 for Agentic Applications 2026 (genai.owasp.org); NIST NICE Framework SP 800-181 Rev.1; NSA CAE-CO program guidance.

Pedagogy. Schneier “The Security Mindset” (2008); Ceraolo et al. *Journal of Cybersecurity* 2023; Chapman, Burket & Brumley “picoCTF” (3GSE 2014); Nelson & Shoshitaishvili “PWN The Learning Curve” (SIGCSE 2024); Vykopal et al. “Benefits and Pitfalls of CTFs” (SIGCSE 2020); Weiss et al. “Reflective Approach to Assessing” (SIGCSE 2016); Carlini “On Evaluating Adversarial Robustness” (arXiv:1902.06705).

Practitioner canon. Johann Rehberger’s *Embrace The Red* (embracethered.com), the primary source for agent exploitation demonstrations (Devin, Cross-Agent Privilege Escalation, ASCII Smuggler, Scary Agent Skills, Gemini Memory Attack); Invariant Labs “MCP

Tool Poisoning” (April 2025); CyberArk Labs “Poison Everywhere”; Unit 42 “MCP Attack Vectors”; Elastic Security Labs “MCP Tools”; authzed.com MCP breach timeline.

Tooling references. Lakera Gandalf (gandalf.lakera.ai); HackAPrompt paper (Schulhoff et al., EMNLP 2023, arXiv:2311.16119); Tensor Trust paper (Toyer et al., ICLR 2024, arXiv:2311.01011); pwn.college (pwn.college); picoCTF (picoctf.org); Microsoft AI Red Teaming Playground Labs (github.com/microsoft/AI-Red-Teaming-Playground-Labs); PyRIT (github.com/microsoft/PyRIT); NVIDIA garak; Promptfoo; NeMo Guardrails; Llama Guard 4 / Prompt Guard 2.

Conclusion, the curriculum's defining thesis

Three claims consolidate everything above.

The instruction-vs-data confusion is a unifying pedagogical spine because the pattern is real, the lineage is documented, and the 2024-2026 defensive research literally names its classical ancestors. CaMeL cites Abadi on CFI, Anderson on access control, and Denning on information flow. Prompt Flow Integrity names CFI. Intent Capsules generalize signed control-flow manifests. Least-Agency extends least privilege. Dual-LLM is W^X at the agent layer. Tool-chaining is ROP. Template injection in LangChain is literally SSTI. A course built on this spine gives students a single conceptual framework that carries them from `gets()` through Rehberger's *Agent Commander*, with each step reinforcing the previous.

But the analogy breaks in exactly the places students most need to understand, and teaching it as "SQL injection for AI" without teaching where it breaks is pedagogical malpractice. Natural language has no parameterization primitive. Transformer attention has no hardware W^X. LLM execution is probabilistic, not deterministic. Adaptive attacks (AMS, October 2025) have broken every training-based defense at ≥90% attack success. The SQL-injection analogy is true enough to motivate the field and false enough to mislead the fix, and students who leave a course believing that "we just need better filters" will build exploitable systems. The curriculum's job is to produce students who instead believe that *structural containment* (capabilities, intent capsules, dual-LLM, rule-of-two) is the only path to meaningful security guarantees, and that even there, the guarantees apply only over restricted task classes.

The tooling and pedagogy are mature enough to execute this curriculum now. picoCTF + Gandalf at high school, pwn.college + HackAPrompt 2.0 + AgentDojo at undergraduate, CaMeL-reimplementation + JailbreakBench submission + MCP-protocol hardening research at graduate, every module has an open-source lab, a peer-reviewed reading, and an assessment rubric. The boundary between classical memory corruption and agentic AI security is where the next generation of security researchers will need to operate, and the architectural parallel is the scaffolding that lets them bring fifty years of systems-security wisdom with them across that boundary, while learning, honestly and explicitly, exactly where that wisdom does not transfer.

References

The following list consolidates citations referenced inline throughout the report. The Annotated Bibliography section above provides editorial commentary on the must-read subset; this list is the full bibliographic register sorted alphabetically by first author. arXiv preprints are cited by their identifier; published venues are named where a peer-reviewed version exists.

Classical systems-security canon

- Abadi, M., Budiu, M., Erlingsson, Ú., & Ligatti, J. (2005). Control-flow integrity. *Proceedings of the 12th ACM Conference on Computer and Communications Security (CCS '05)*.
- Carlini, N., & Wagner, D. (2015). Control-flow bending: On the effectiveness of control-flow integrity. *USENIX Security 2015*.
- Denning, D. E. (1976). A lattice model of secure information flow. *Communications of the ACM*, 19(5), 236-243. https://doi.org/10.1145/360051.360056
- Dennis, J. B., & Van Horn, E. C. (1966). Programming semantics for multiprogrammed computations. *Communications of the ACM*, 9(3), 143-155.
- Hardy, N. (1988). The Confused Deputy (or why capabilities might have been invented). *ACM SIGOPS Operating Systems Review*, 22(4), 36-38. https://doi.org/10.1145/54289.871709
- Miller, M. S. (2006). *Robust composition: Towards a unified approach to access control and concurrency control* (Doctoral dissertation). Johns Hopkins University. http://erights.org/talks/thesis/markm-thesis.pdf
- Myers, A. C. (1999). JFlow: Practical mostly-static information flow control. *POPL '99*.
- Rajani, V., Garg, D., & Rezk, T. (2016). On access control, capabilities, their equivalence, and confused deputy attacks. *IEEE Computer Security Foundations Symposium (CSF 2016)*. https://people.mpi-sws.org/~dg/papers/csf16-caps.pdf
- Sabelfeld, A., & Myers, A. C. (2003). Language-based information-flow security. *IEEE Journal on Selected Areas in Communications*, 21(1), 5-19.
- Saltzer, J. H., & Schroeder, M. D. (1975). The protection of information in computer systems. *Proceedings of the IEEE*, 63(9), 1278-1308. https://www.cs.virginia.edu/~evans/cs551/saltzer/
- Shacham, H. (2007). The geometry of innocent flesh on the bone: Return-into-libc without function calls (on the x86). *CCS '07*.
- Shoshitaishvili, Y., Wang, R., Salls, C., et al. (2016). SoK: (State of) the art of war: Offensive techniques in binary analysis. *IEEE S&P 2016*.

Foundational agentic-security primary sources

- 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. *AISec 2023*. arXiv:2302.12173

- Wei, A., Haghtalab, N., & Steinhardt, J. (2023). Jailbroken: How does LLM safety training fail? arXiv:2307.02483
- Willison, S. (2022, September 12). Prompt injection attacks against GPT-3. *simonwillison.net*. https://simonwillison.net/2022/Sep/12/prompt-injection/
- Willison, S. (2023, April 25). The Dual LLM pattern for building AI assistants that can resist prompt injection. *simonwillison.net*. https://simonwillison.net/2023/Apr/25/dual-llm-pattern/
- Willison, S. (2025, April). MCP has prompt injection security problems. *simonwillison.net*.
- Willison, S. (2025, June 13). Design patterns for securing LLM agents against prompt injection (review). *simonwillison.net*. https://simonwillison.net/2025/Jun/13/
- Willison, S. (2025, June 16). The lethal trifecta for AI agents. *simonwillison.net*. https://simonwillison.net/2025/Jun/16/the-lethal-trifecta/
- 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

2024-2026 architectural and model-level defenses

- Anthropic. (2025). Constitutional Classifiers: Defending against universal jailbreaks across thousands of hours of red teaming. arXiv:2501.18837
- Anthropic. (2026). Constitutional Classifiers++. arXiv:2601.04603
- Beurer-Kellner, L., et al. (2025). Design patterns for securing LLM agents against prompt injection. arXiv:2506.08837
- Beurer-Kellner, L., et al. (2025). [Plan-then-execute as control-flow integrity for indirect prompt injection.] arXiv:2509.08646
- Cao, B., et al. (2023). Defending against alignment-breaking attacks via robustly aligned LLM. arXiv:2309.14348
- Chen, S., et al. (2024). StruQ: Defending against prompt injection with structured queries. arXiv:2402.06363. *USENIX Security 2025*.
- Chen, S., et al. (2024). Aligning LLMs to be robust against prompt injection (SecAlign). arXiv:2410.05451. *CCS 2025*.
- Debenedetti, E., et al. (2025). Defeating prompt injections by design (CaMeL). arXiv:2503.18813
- Hines, K., Lopez, G., Hall, M., Zarfati, F., Zunger, Y., & Kiciman, E. (2024). Defending against indirect prompt injection attacks with spotlighting. arXiv:2403.14720
- Hu, J., et al. (2024). SemanticSmooth: Smoothed text classifier-based jailbreak defense. arXiv:2402.16192
- Kim, J., Song, D., et al. (2025). Prompt Flow Integrity to prevent privilege escalation in LLM agents. arXiv:2503.15547
- Meta. (2025). Meta-SecAlign: Frontier-grade defended language models for indirect prompt injection. arXiv:2507.02735
- Robey, A., Wong, E., Hassani, H., & Pappas, G. J. (2023). SmoothLLM: Defending large language models against jailbreaking attacks. arXiv:2310.03684
- Wallace, E., Xiao, K., Leike, J., et al. (2024). The instruction hierarchy: Training LLMs to prioritize privileged instructions. arXiv:2404.13208
- Wang, H., et al. (2025). AgentSpec: Customizable runtime enforcement for safe and reliable LLM agents. arXiv:2503.18666

- Zhu, Y., et al. (2025). MELON: Provable defense against prompt injection. arXiv:2502.05174. *ICML 2025*.

Critical evaluation and adaptive attacks

- Carlini, N., & Wagner, D. (2019). On evaluating adversarial robustness. arXiv:1902.06705
- Carlini, N. (2024). [Are aligned neural networks adversarially aligned?]. *NeurIPS 2024*.
- Nasr, M., Carlini, N., Tramèr, F., et al. (2025). The attacker moves second: Stronger adaptive attacks bypass defenses against LLM jailbreaks and prompt injections. arXiv:2510.09023
- Wu, T., et al. (2025). Failure of instruction hierarchies under mutually exclusive constraints. arXiv:2502.15851

Attacks on agents, tools, and memory

- Anthropic. (2024, April). Many-shot jailbreaking. *Anthropic Research*.
- Anthropic, UK AISI, & Alan Turing Institute. (2025). [Constant-cost backdoor study: 250 malicious documents suffice across model scale.] arXiv:2510.07192
- CorruptRAG. (2025). arXiv:2504.03957
- *Plentiful Jailbreaks with String Compositions*. (2024). arXiv:2411.01084
- Schroeder de Witt, C., et al. (2025). Open challenges in multi-agent security. arXiv:2505.02077
- Shah, R., et al. (2023). Scalable and transferable black-box jailbreaks for language models via persona modulation. arXiv:2311.03348
- *StructuralSleight*. (2024). arXiv:2406.08754
- Zeng, Y., Lin, H., Zhang, J., et al. (2024). How Johnny can persuade LLMs to jailbreak them: Rethinking persuasion to challenge AI safety by humanizing LLMs. *ACL 2024*. arXiv:2401.06373
- Zou, W., Geng, R., Wang, B., & Jia, J. (2024). PoisonedRAG: Knowledge corruption attacks to retrieval-augmented generation of large language models. arXiv:2402.07867

MCP protocol attacks and defenses

- *BlindGuard*. (2025). arXiv:2508.08127
- BlockAgents. (2024). *ACM TCCE 2024*.
- *Dark Side of LLMs: Peer-agent injection susceptibility*. (2025). arXiv:2507.06850
- Invariant Labs. (2025, April). MCP tool poisoning attacks. *invariantlabs.ai*.
- Maloyan, & Namiot. (2026). [Protocol-level gaps in MCP.] arXiv:2601.17549
- *MCPLib: 31 attacks across four classes including Multi-Tool Cooperation Attack*. (2025). arXiv:2508.12538
- *MCPTox: 1,312 malicious cases across 11 categories*. (2025). arXiv:2508.14925
- Rehberger, J. (2025-2026). *Embrace The Red* (blog). embracethered.com, including: “AI Kill Chain in Action: Devin AI Exposes Ports” (Aug 2025); “Cross-Agent Privilege Escala-

tion: When Agents Free Each Other" (2025); "Agent Commander: Your Agent Works For Me Now" (2026); ASCII Smuggler; Gemini Memory Attack.
- *When MCP servers attack: Control-flow hijack via tool return values*. (2025). arXiv:2509.24272

Benchmarks, testbeds, and red-teaming frameworks

- Andriushchenko, M., et al. (2025). AgentHarm: A benchmark for measuring harmfulness of LLM agents. *ICLR 2025*.
- Center for AI Safety. (2024). HarmBench: A standardized evaluation framework for automated red teaming and robust refusal. *ICML 2024*.
- Chao, P., et al. (2024). JailbreakBench: An open robustness benchmark for jailbreaking large language models. *NeurIPS 2024 Datasets & Benchmarks*.
- Debenedetti, E., et al. (2024). AgentDojo: A dynamic environment to evaluate prompt injection attacks and defenses for LLM agents. arXiv:2406.13352. *NeurIPS 2024 Datasets & Benchmarks*.
- *Do-Not-Answer*. (2024). *EACL 2024* (LibrAI / University of Melbourne).
- Microsoft. (2025). BIPIA: Benchmark for indirect prompt injection attacks. *KDD 2025*.
- Schulhoff, S., et al. (2023). HackAPrompt: Exposing systemic vulnerabilities of LLMs through a global prompt hacking competition. *EMNLP 2023* (Best Theme Paper). arXiv:2311.16119
- Toyer, S., et al. (2024). Tensor Trust: Interpretable prompt injection attacks from an online game. *ICLR 2024*. arXiv:2311.01011
- *TrustLLM: Trustworthiness in large language models* (eight-dimensional benchmark across 30+ datasets). (2024).
- Zhan, Q., et al. (2024). InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents. *ACL 2024 Findings*.
- Zhang, H., et al. (2025). Agent Security Bench (ASB): Formalizing and benchmarking attacks and defenses in LLM-based agents. *ICLR 2025*.

Standards, governance, and policy

- Alan Turing Institute CETaS. (2024, November). *Indirect prompt injection: Generative AI's greatest security flaw* (Ruck & Sutton).
- Auth0. (2025). [Authorization for AI agents: Never let the agent's chat surface grant permission.] *auth0.com/blog*.
- Meta. (2025). Agents Rule of Two. *Meta AI*.
- NCSC (UK National Cyber Security Centre). *Prompt injection is not SQL injection (it may be worse)*. https://www.ncsc.gov.uk
- NIST. (2025). NICE framework: Workforce framework for cybersecurity (SP 800-181 Rev.1).
- NSA. CAE-CO program guidance.
- OpenAI. (2025). Approach to external red teaming for AI models and systems.
- OpenAI. (2025). *Model Spec*.

- OWASP GenAI Security Project, Agentic Security Initiative (Sotiropoulos, Katz, Del Rosario, Habler, et al.). (2025, December 9). *OWASP Top 10 for Agentic Applications 2026*. https://genai.owasp.org
- Vassilev, A., et al. (2025, March). NIST AI 100-2 E2025: Adversarial machine learning, A taxonomy and terminology of attacks and mitigations.

Pedagogy and curriculum design

- Carlini, N. (n.d.). Reading lists on adversarial machine learning. https://nicholas.carlini.com
- Ceraolo, A., et al. (2023). [Empirical study of adversarial-thinking development.] *Journal of Cybersecurity*.
- Chapman, P., Burket, J., & Brumley, D. (2014). PicoCTF: A game-based computer security competition for high school students. *3GSE 2014*.
- Nelson, C., & Shoshitaishvili, Y. (2024). PWN the learning curve: Education-first CTF challenges. *SIGCSE 2024*.
- O'Leary, M. (2017). [CTF-based pedagogy.] *SIGCSE 2017*.
- Schneier, B. (2008). The security mindset. *Schneier on Security* blog.
- Vykopal, J., et al. (2020). Benefits and pitfalls of using capture the flag games in university courses. *SIGCSE 2020*.
- Weiss, R., Mache, J., et al. (2016). Reflective approach to assessing CTF-based programs. *SIGCSE 2016*.

Tooling, testbeds, and labs (URLs)

- AgentDojo. https://agentdojo.spylab.ai
- E2B (Firecracker microVMs). https://e2b.dev
- Gandalf (Lakera). https://gandalf.lakera.ai
- HackAPrompt 2.0. https://hackaprompt.com
- mcp-scan / Snyk agent-scan (Invariant Labs). https://github.com/invariantlabs-ai/mcp-scan
- Microsoft AI Red Teaming Playground Labs. https://github.com/microsoft/AI-Red-Teaming-Playground-Labs
- NVIDIA garak (Apache 2.0). https://github.com/leondz/garak
- NVIDIA NeMo Guardrails. https://github.com/NVIDIA/NeMo-Guardrails
- picoCTF. https://picoctf.org
- Promptfoo (MIT). https://promptfoo.dev
- Protect AI LLM Guard. https://github.com/protectai/llm-guard
- pwn.college. https://pwn.college
- PyRIT (Microsoft). https://github.com/microsoft/PyRIT
- Tensor Trust. https://tensortrust.ai

CVEs and incident references

- CVE-2025-32711, Microsoft Copilot zero-click email exfiltration (“EchoLeak”).
- CVE-2025-65106, LangChain Jinja2 template injection (attribute-access).
- CVE-2025-68664, LangChain Core serialization SSTI (“LangGrinch”, CVSS 9.3).
- CVE-2025-9556, LangChainGo / Gonja SSTI (CVSS 9.8).
- CVE-2025-6514, OS command injection in mcp-remote.