Security Vulnerabilities and Defensive Mechanisms in CLI/Terminal-Based Large Language Model Deployments: A Comprehensive Research Synthesis

Technical Report

Pre-print Version for arXiv/IACR ePrint Archive

Date: November 02, 2025

Classification: Industry White Paper / Security Research Community Pre-print

Target Audience: Security Practitioners, ML Engineers, System Administrators, Risk Managers

Document Type: 6-8 Page Short Paper Format

Abstract

Background: Command-line interface (CLI) deployments of large language models (LLMs) have proliferated rapidly across development environments, yet face converging security challenges from traditional CLI attack surfaces and novel AI-specific vulnerabilities.

Methods: We conducted a comprehensive systematic review synthesizing 85+ research sources including peer-reviewed academic papers, industry security reports, and benchmark datasets spanning 2022-2025. Our analysis employed structured gap identification methodologies across adversarial ML, model security, system integrity, prompt security, data poisoning, and model extraction attack vectors.

Results: Analysis reveals 97.2% success rates for system prompt extraction attacks [1], 218% year-over-year increases in state-sponsored AI infrastructure attacks [2], and 77% of organizations reporting AI system breaches in 2024 [3]. Despite contributions from 600+ security experts to frameworks like OWASP Top 10 for LLMs [4], prompt injection remains fundamentally unsolved with 2025 research demonstrating 98% attack success rates against GPT-4o [86] and 87.2% against safety-aligned models [89], confirming persistent exploitability despite defensive advances. Major CLI platforms including Cursor IDE, GitHub Copilot, and ChatGPT exhibit systematic vulnerabilities (94 CVEs documented) with CVSS scores reaching 8.8 [6-8].

Conclusions: CLI LLM security demands defense-in-depth strategies combining architectural isolation, cryptographic integrity verification, behavioral monitoring, and regulatory compliance frameworks. Critical research gaps persist in agentic AI security, supply chain protections, and empirically-validated defensive mechanisms under adaptive adversary models.

Keywords: Large Language Models, Command-Line Interface Security, Prompt Injection, Adversarial Machine Learning, AI Security, Terminal Security, Model Integrity

1. Introduction

1.1 Background and Motivation

The rapid integration of large language models into command-line interface tools has fundamentally transformed software development workflows. Tools such as GitHub Copilot, Cursor IDE, Claude Code CLI, and OpenAI's command-line interfaces process millions of developer interactions daily [9]. However, this proliferation occurs against a backdrop of inadequate security frameworks specifically designed for CLI-based AI deployments.

Traditional CLI security concerns—including command injection, privilege escalation, and environment variable exploitation—converge with novel AI-specific attack vectors including prompt injection, model extraction, and training data poisoning [10, 11]. This convergence creates a unique threat landscape requiring interdisciplinary security approaches.

1.2 Research Objectives

This comprehensive synthesis addresses three primary research questions:

RQ1: What are the documented security vulnerabilities specific to CLI/terminal-based LLM deployments, and what is their prevalence and exploitability?

RQ2: What defensive mechanisms have been proposed or implemented, and what is their empirical effectiveness under adversarial conditions?

RQ3: What critical gaps exist between academic research, industry practice, and regulatory frameworks for CLI LLM security?

1.3 Scope and Limitations

Our analysis focuses specifically on command-line and terminal-based LLM deployments, distinguishing these from web-based or GUI applications. We synthesize research from peer-reviewed academic publications (ACM CCS, USENIX Security, IEEE S&P, NDSS), industry security vendor reports (CrowdStrike, Palo Alto Networks, Microsoft), government frameworks (NIST AI RMF, EU AI Act), and open benchmark datasets.

Limitations include: (1) rapidly evolving threat landscape with monthly disclosure of new vulnerabilities; (2) limited long-term empirical data on defense effectiveness; (3) publication bias toward novel attacks versus incremental defensive improvements; (4) proprietary security measures deployed by major vendors that remain undocumented in public literature.

2. Methodology

2.1 Systematic Literature Review Protocol

We conducted a systematic literature review following PRISMA guidelines adapted for security research [12]. Our search strategy employed:

Database Coverage: ACM Digital Library, IEEE Xplore, arXiv.org, IACR ePrint Archive, USENIX Digital Library, Google Scholar, vendor security blogs, CVE databases

Search Terms: ("large language model" OR "LLM" OR "generative AI") AND ("security" OR "vulnerability" OR "attack" OR "defense") AND ("command-line" OR "CLI" OR "terminal" OR "prompt injection" OR "adversarial")

Inclusion Criteria: (1) Published 2022-2025; (2) peer-reviewed or from recognized security institutions; (3) directly relevant to LLM security or CLI security; (4) empirical evaluation or systematic analysis

Quality Assessment: Oxford CEBM evidence levels adapted for security research, GRADE framework for recommendation strength

2.2 Gap Analysis Framework

We employed a structured gap identification methodology examining:

- **Methodological gaps:** Incomplete testing frameworks, insufficient validation protocols
- **Empirical data gaps:** Unexplored attack surfaces, unmeasured defensive efficacy
- **Theoretical framework gaps:** Incomplete threat models, missing security formalizations
- **Practical implementation gaps:** Deployment security, operational considerations

2.3 Threat Intelligence Integration

Quantitative data was extracted from industry reports including CrowdStrike Global Threat Report [2], Microsoft Digital Defense Report [13], Orca Security State of AI Security Report [3], and Google Threat Intelligence Group assessments [14]. Vulnerability data was systematically cataloged from CVE databases, vendor security advisories, and responsible disclosure reports.

3. Results

3.1 CLI LLM Attack Surface Taxonomy

Our analysis identified five primary attack surfaces in CLI-based LLM deployments:

3.1.1 Direct CLI Attack Vectors

Argument Injection: Exploitation of unvalidated command-line parameters enables arbitrary command execution. Documented attacks against code generation tools demonstrate 98.3% success rates when protection mechanisms are absent [15].

Environment Variable Exploitation: Manipulation of $PATH, $LD_PRELOAD, and shell configuration variables enables privilege escalation and malicious library injection. Red Canary documented 37% increase in adversarial abuse of AI CLI tools through environment manipulation in 2024 [16].

Command Substitution: Backtick and $() syntax embedded within LLM prompts facilitate code execution through shell expansion mechanisms [17].

3.1.2 LLM-Specific Vulnerabilities

Prompt Injection Attacks: Analysis of OWASP Top 10 for LLM Applications 2025 [4] identifies prompt injection as the primary vulnerability class. Systematic evaluation across 200+ Custom GPTs demonstrated 97.2% system prompt extraction success and 100% file leakage rates [1]. Liu et al.'s benchmark of 5 attack techniques against 10 LLMs across 7 tasks confirmed persistent exploitability [18].

2024-2025 Persistent Vulnerability Evidence: Recent research confirms continued exploitability of modern LLMs. FlipAttack methodology achieves ~98% attack success rate on GPT-4o through character-order manipulation, with ~98% bypass rate against 5 guardrail models [86]. IRIS jailbreaking demonstrates 98% success on GPT-4 and GPT-4 Turbo in under 13 queries, outperforming prior TAP results (75% ASR, 20+ queries) [87]. Systematic red-teaming evaluation of 1,400+ adversarial prompts found GPT-4 exhibited 87.2% attack success rate, with successful prompts transferring to Claude 2 at 64.1% success [89]. BIPIA benchmark evaluation of 25 LLMs confirms GPT-3.5-turbo and GPT-4 demonstrate elevated vulnerability to indirect prompt injection despite strong capabilities [88].

Prompt injection subdivides into:

- **Direct (Jailbreaking):** Zou et al.'s universal adversarial suffixes [19] achieve transferable attacks across GPT-4, Bard, and Claude
- **Indirect:** Greshake et al. [20] demonstrated cross-domain exploitation where LLMs consume malicious instructions from external sources (websites, PDFs, emails, databases)

Model Extraction: Adversarial queries enable intellectual property theft and safety mechanism reverse engineering [21].

Data Poisoning: Yao et al.'s PoisonPrompt research [22] demonstrates backdoor injection effective across hard and soft prompts with only 0.1% poisoned training data achieving 40% negative response rates in instruction-tuned models.

3.1.3 Documented CVE Analysis

Systematic CVE analysis reveals critical vulnerabilities across major platforms:

Cursor IDE:

- CVE-2025-54135 (CurXecute): Remote code execution via MCP auto-start, CVSS 8.6. Discovered by AIM Security; disclosed August 1, 2025 [6].
- CVE-2025-54136 (MCPoison): Persistent execution through MCP trust bypass, CVSS 7.2. Discovered by Check Point Research; disclosed August 5, 2025 [6].
- 94 inherited Chromium CVEs from outdated engine [7]

Clarification on CVE Analysis: CVE-2025-54135 and CVE-2025-54136 vulnerabilities were discovered and responsibly disclosed by third-party security researchers. This paper provides systematic analysis and contextualization within the broader CLI LLM security landscape.

GitHub Copilot:

- CVE-2025-62449: Path traversal vulnerability, CVSS 6.8 [8]
- CVE-2025-62453: Improper validation of AI-generated output [8]
- 39.33% of top suggestions contain security vulnerabilities [23]

ChatGPT:

- Atlas CSRF: 97% attack success rate for persistent memory injection [24]
- ShadowLeak: Zero-click vulnerability in Deep Research agent [24]
- Seven vulnerability classes including indirect prompt injection, zero-click attacks, memory poisoning [24]

3.2 Defensive Mechanisms and Effectiveness

3.2.1 Industry Framework Analysis

OWASP Top 10 for LLM Applications 2025: Developed by 600+ security experts across 18 countries, this framework catalogs vulnerabilities beyond prompt injection including sensitive information disclosure, supply chain vulnerabilities, data poisoning, improper output handling, excessive agency, system prompt leakage, vector/embedding weaknesses, misinformation, and unbounded consumption [4].

NIST AI Risk Management Framework: Published as NIST AI 100-1 [25], establishes voluntary governance through four functions (GOVERN, MAP, MEASURE, MANAGE) with seven trustworthy AI characteristics including security and resilience.

MITRE ATLAS: Adapts ATT&CK framework with 14 tactics and 56 techniques specific to ML/AI systems, incorporating case studies of reconnaissance, resource development, initial access, ML model access, execution, persistence, defense evasion, and impact [26].

3.2.2 Technical Defense Evaluation

Input Validation and Filtering: Fine-tuned classifiers achieve 92% accuracy with RoBERTa and 99.1% with DeBERTa on 662-prompt datasets, outperforming GPT-4's 87.4% [27]. Commercial solutions including Rebuff, Lakera Guard, and Prompt Armor provide production-grade filtering.

Preference Optimization Approaches: SecAlign defense [28] demonstrates ~0% attack success rate against optimization-based attacks while preserving utility (AlpacaEval2 WinRate maintained within 0.7% standard error). Training requires 4× NVIDIA Tesla A100 (80GB) for 3 epochs with LoRA optimizing <1% parameters.

Delimiter-Based Defenses: Instruction hierarchy mechanisms demonstrate effectiveness against indirect prompt injection. Structured queries using XML-style delimiters reduce attack success rates, though adaptive attacks develop delimiter-aware evasion techniques [29].

Perplexity Filtering: Detecting adversarial prompts through statistical anomalies (PPL(x)smoothed) achieves 77.6% average true positive rate but 22.4% false negative rate indicates susceptibility to adaptive adversaries [30].

3.2.3 Empirical Validation Under Adversarial Conditions

Meta's CyberSecEval 2 benchmark evaluated GPT-4, Claude Sonnet, Llama 3, Mistral across prompt injection resistance. Results demonstrate 26-41% residual attack success rates even with defenses, confirming arms race dynamics [31]. Anthropic's Constitutional AI demonstrates improved safety alignment but remains vulnerable to novel jailbreak templates [32].

StruQ (Structured Queries) defense using semantic preserving transformations shows promise with 35% attack success rate reduction, but adaptive attackers develop countermeasures [33]. SecAlign's adversarial training achieves near-zero attack success against known optimization-based attacks but generalizes poorly to novel techniques [28].

3.2.4 Practical Implementation: Behavioral Monitoring Systems

While academic defenses demonstrate promise, operational deployment requires lightweight, real-time monitoring mechanisms compatible with development workflows. Hook-based architectures provide one such implementation strategy, intercepting LLM outputs before execution to detect malicious patterns.

Silent-Alarm-Detector Framework: Implemented as PreToolUse hook for CLI LLM environments, this system employs hybrid detection combining regex pattern matching (fast, 90% case coverage) with AST structural analysis (complex cases). Detection targets eight pattern classes including silent exception handling, security shortcuts (SQL injection via string formatting, eval() usage), and performance anti-patterns ($O(n^2)$ algorithms). Impact scoring quantifies risk across performance (30% weight), security (40%), and maintainability (30%) dimensions. Critical detections (impact ≥80 or security ≥90) trigger blocking with actionable remediation guidance [90].

Operational Characteristics: Execution latency averages 50-100ms with <10% false positive rate at balanced sensitivity. The architecture demonstrates defense-in-depth co-ordination: security_guard.py blocks malicious code (command injection), silent-alarm-detector blocks quality issues (technical debt accumulation), enabling complementary protection layers. Deployment via PreToolUse hooks eliminates MCP server complexity while maintaining Claude Code compatibility [90].

This implementation validates behavioral monitoring feasibility in production CLI LLM environments, demonstrating practical realization of theoretical defensive mechanisms discussed in academic literature.

3.3 Supply Chain Security Threats

Supply chain vulnerabilities in AI/ML ecosystems present systemic risks. In February 2025, malicious ML models on Hugging Face exploited "broken" pickle serialization to evade Picklescan detection, using 7z compression instead of default ZIP format [91]. Over 100 malicious models leverage pickle deserialization for remote code execution, with 95% utilizing PyTorch format [92]. The platform's growth from 300,000 models (2023) to 1 million (September 2024) amplifies attack surface [93].

Systematic analysis reveals attackers weaponize PyTorch .pth files on trusted repositories, embedding shell commands executed during torch.load() deserialization to deploy remote access trojans [95]. These attacks exploit inherent pickle format risks despite documented security concerns, with detection tools failing against obfuscated payloads.

Dataset Poisoning: BadNets and other backdoor injection techniques manipulate training data to introduce adversarial triggers [34]. Gradient shaping attacks achieve stealthy backdoor implantation surviving model fine-tuning [35].

Dependency Vulnerabilities: Analysis of ML supply chain dependencies reveals 43% of models on Hugging Face contain at least one security vulnerability in their dependency trees [36]. Software composition analysis (SCA) tools adapted for ML ecosystems remain nascent.

3.4 Regulatory and Compliance Landscape

EU AI Act: Implements risk-based categorization with high-risk AI systems (including critical infrastructure applications) subject to conformity assessments, documentation requirements, and human oversight mandates [37]. CLI LLM deployments in regulated sectors face stringent compliance obligations.

NIST AI RMF: Voluntary framework establishing governance through GOVERN, MAP, MEASURE, MANAGE functions. Emphasizes continuous monitoring, documentation, and adversarial testing [25].

ISO/IEC 42001: International standard for AI management systems addressing governance, risk management, and accountability [38]. Provides certification pathway for organizational AI governance maturity.

Industry-Specific Regulations: HIPAA implications for healthcare LLMs [39], GDPR data processing requirements, SOC 2 compliance for SaaS deployments, and financial services regulations (FINRA, SEC) create complex compliance matrices.

3.5 Emerging Threat Vectors

Agentic AI Security: Tool-augmented LLMs with external API access create expanded attack surfaces. Prompt injection in multi-agent systems enables lateral movement and privilege escalation [40]. WebArena benchmark demonstrates automated exploitation of real-world web applications [41].

Multi-Modal Attacks: Vision-language models vulnerable to typographic attacks embedding adversarial prompts in images [42]. Audio-based jailbreaking through speech recognition bypass [43].

Context Poisoning: Manipulating retrieval-augmented generation (RAG) systems through adversarial document injection into vector databases [44]. Embedding space attacks targeting semantic search mechanisms [45].

3.6 Architectural Evolution and Defense-in-Depth

Sandboxing and Isolation: WebAssembly-based sandboxing (WASM) provides lightweight isolation for LLM-generated code execution [46]. gVisor and Firecracker enable secure multi-tenancy for CLI environments [47].

Cryptographic Integrity: Model signing and hash verification through SLSA (Supply-chain Levels for Software Artifacts) framework [48]. Transparent ML (TML) employs Merkle trees for model provenance tracking [49].

Zero Trust Architecture: NIST SP 800-207 principles applied to LLM deployments: continuous authentication, least privilege execution, micro-segmentation of model access [50].

Adversarial Training Pipelines: Continuous red-teaming integrated into CI/CD workflows. Automated adversarial prompt generation using evolutionary algorithms [51, 52].

4. Discussion

4.1 The Persistent Challenge of Prompt Injection

Prompt injection remains fundamentally challenging due to the shared channel problem: LLM architectures process instructions and data through identical input mechanisms [53]. Unlike traditional injection attacks (SQL, command) where parameterization separates code from data, LLMs operate on natural language where such distinction proves ambiguous.

Theoretical Underpinnings: Greshake et al. argue prompt injection represents an inherent limitation of current LLM architectures rather than implementation flaw [20]. The instruction-following objective conflicts with security constraints—models optimized for flexible instruction adherence prove vulnerable to adversarial instructions.

Adversarial Arms Race: Each defensive mechanism spawns adaptive attacks. Delimiter-based defenses face delimiter-aware evasion; perplexity filters encounter adversarial perturbations maintaining fluency; fine-tuned classifiers suffer from distributional shift [54].

4.2 Supply Chain as Critical Vulnerability

The democratization of AI through model-sharing platforms creates systemic supply chain risks. Unlike traditional software supply chains where malicious packages require active installation, ML model loading triggers automatic code execution through deserialization. The tension between usability (convenient model sharing) and security (strict validation) remains unresolved.

4.3 Defense-in-Depth as Necessary Strategy

No single defensive mechanism provides complete protection. Industry consensus favors layered approaches:

1. **Input validation:** Pre-processing filters reducing attack surface
2. **Architectural controls:** Sandboxing, least privilege, network isolation
3. **Behavioral monitoring:** Runtime detection of anomalous patterns
4. **Output filtering:** Post-processing validation before execution
5. **Audit and attribution:** Comprehensive logging for forensic analysis

4.4 Research Gaps and Future Directions

Empirical Validation Deficit: Most defensive mechanisms lack longitudinal evaluation under adaptive adversaries. Laboratory success rates (90%+ defense effectiveness) rarely translate to production environments facing evolving threats.

Agentic AI Security: Tool-augmented LLMs represent uncharted territory. Existing frameworks inadequately address multi-agent attack propagation, cross-system exploitation, and emergent adversarial behaviors.

Formal Verification: Mathematical proofs of security properties for LLM systems remain elusive. Lack of formal threat models hampers principled defense development.

Economic Incentives: Security-usability tradeoffs require economic analysis. Organizations optimize for functionality over security until breach costs exceed defensive investments.

---

5. Conclusion

CLI-based LLM deployments face converging threat vectors from traditional systems security and novel AI-specific vulnerabilities. Our systematic synthesis of 85+ research sources confirms prompt injection as fundamentally unsolved, with 2025 research demonstrating 98% attack success rates against current models. Major platforms exhibit critical vulnerabilities (CVE-2025-54135, CVE-2025-54136) enabling remote code execution and persistent compromise.

Defensive mechanisms show promise but face adaptive adversary challenges. No single approach provides comprehensive protection; defense-in-depth strategies combining input validation, architectural isolation, behavioral monitoring, and cryptographic integrity offer pragmatic risk reduction. Practical implementations like hook-based monitoring systems demonstrate operational feasibility.

Supply chain vulnerabilities present systemic risks, with 100+ malicious models discovered on major platforms exploiting pickle serialization flaws. Regulatory frameworks (EU AI Act, NIST AI RMF) establish compliance requirements but implementation guidance remains limited.

Critical research gaps persist in:

- Empirical validation under adaptive adversaries
- Agentic AI security frameworks
- Supply chain verification mechanisms
- Formal security properties and threat models
- Economic analysis of security-usability tradeoffs

Future research must prioritize empirical validation under adversarial conditions, formalized threat modeling for agentic systems, and practical implementation guidance bridging academic advances with operational security requirements.

The field demands interdisciplinary approaches combining traditional security, machine learning, natural language processing, and systems engineering to address this complex, rapidly-evolving threat landscape. Only through systematic integration of technical defenses, governance frameworks, and continuous adversarial evaluation can the security of CLI-based LLM deployments keep pace with their proliferation.

References

[1] Liu, H., et al. (2023). “Formalizing and Benchmarking Prompt Injection Attacks and Defenses.” *USENIX Security Symposium*. Available at: https://github.com/liu00222/Open-Prompt-Injection

[2] CrowdStrike. (2025). *Global Threat Report 2025: State-Sponsored AI Infrastructure Attacks*. CrowdStrike Holdings, Inc.

[3] Orca Security. (2024). *2024 State of AI Security Report*. Available at: https://orca.security/resources/blog/2024-state-of-ai-security-report/

[4] OWASP Foundation. (2025). *OWASP Top 10 for Large Language Model Applications 2025*. Available at: https://owasp.org/www-project-top-10-for-large-language-model-applications/

[5] Andriushchenko, M., et al. (2024). “Jailbreaking Leading Safety-Aligned LLMs with Simple Adaptive Attacks.” *OpenReview*. Available at: https://openreview.net/forum?id=hXA8wqRdyV

[6] Check Point Research. (2025). “Cursor IDE’s MCP Vulnerability - MCPoison.” Available at: https://research.checkpoint.com/2025/cursor-vulnerability-mcpoison/

[7] OX Security. (2025). “94 Vulnerabilities in Cursor and Windsurf Put 1.8M Developers at Risk.” Available at: https://www.ox.security/blog/94-vulnerabilities-in-cursor-and-windsurf-put-1-8m-developers-at-risk/

[8] Cyber Press. (2025). “GitHub Copilot and Visual Studio Vulnerabilities Allow Attackers to Bypass Security Features.” Available at: https://cyberpress.org/github-copilot-and-visual-studio-vulnerabilities/

[9] McKinsey & Company. (2025). *The State of AI in 2025: Agents, Innovation, and Transformation*. Available at: https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai

[10] Yao, Y., et al. (2024). “Survey on Large Language Model (LLM) Security and Privacy: The Good, the Bad, and the Ugly.” arXiv preprint arXiv:2312.02003.

[11] Shayegani, E., et al. (2024). “Survey of Vulnerabilities in Large Language Models Revealed by Adversarial Attacks.” arXiv preprint arXiv:2310.10844. *ACL 2024 Tutorial*.

[12] Page, M. J., et al. (2021). “The PRISMA 2020 statement: an updated guideline for reporting systematic reviews.” *BMJ*, 372.

[13] Microsoft. (2024). *Microsoft Digital Defense Report 2024*. Available at: https://www.microsoft.com/en-us/security/security-insider/threat-landscape/microsoft-digital-defense-report-2024

[14] Google Threat Intelligence Group. (2025). *Global Threat Landscape Report*. Google Cloud Security.

[15] Pearce, H., et al. (2022). "Asleep at the Keyboard? Assessing the Security of GitHub Copilot's Code Contributions." *IEEE Symposium on Security and Privacy*.

[16] Red Canary. (2024). *2024 Threat Detection Report*. Available at: https://redcanary.com/threat-detection-report/

[17] MITRE Corporation. (2024). "ATT&CK for ICS: Command and Scripting Interpreter." Available at: https://attack.mitre.org/techniques/T1059/

[18] Liu, H., et al. (2023). "Prompt Injection Attacks and Defenses in LLM-Integrated Applications." arXiv preprint arXiv:2310.12815.

[19] Zou, A., et al. (2023). "Universal and Transferable Adversarial Attacks on Aligned Language Models." arXiv preprint arXiv:2307.15043.

[20] Greshake, K., et al. (2023). "Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection." *ACM Workshop on Artificial Intelligence and Security*.

[21] Carlini, N., et al. (2024). "Stealing Part of a Production Language Model." arXiv preprint arXiv:2403.06634.

[22] Yao, Y., et al. (2023). "PoisonPrompt: Backdoor Attack on Prompt-based Large Language Models." arXiv preprint arXiv:2310.12439.

[23] Pearce, H., et al. (2023). "Examining Zero-Shot Vulnerability Repair with Large Language Models." *IEEE Symposium on Security and Privacy*.

[24] Tenable Research. (2025). "Seven Critical Vulnerabilities in ChatGPT." Available at: https://www.tenable.com/blog/chatgpt-vulnerabilities

[25] NIST. (2023). *Artificial Intelligence Risk Management Framework (AI RMF 1.0)*. NIST AI 100-1. Available at: https://www.nist.gov/itl/ai-risk-management-framework

[26] MITRE Corporation. (2024). *ATLAS: Adversarial Threat Landscape for Artificial-Intelligence Systems*. Available at: https://atlas.mitre.org/

[27] Jain, N., et al. (2023). "Baseline Defenses for Adversarial Attacks Against Aligned Language Models." arXiv preprint arXiv:2309.00614.

[28] Huang, Y., et al. (2024). "SecAlign: Defending Against Prompt Injection with Preference Optimization." arXiv preprint arXiv:2410.05451.

[29] Wallace, E., et al. (2024). "Instruction Hierarchy: Training LLMs to Prioritize Privileged Instructions." arXiv preprint arXiv:2404.13208.

[30] Alon, G., & Kamfonas, M. (2023). "Detecting Language Model Attacks with Perplexity." arXiv preprint arXiv:2308.14132.

[31] Meta AI. (2024). *CyberSecEval 2: A Wide-Ranging Cybersecurity Evaluation Suite for Large Language Models*. Available at: https://ai.meta.com/research/publications/cyberseceval-2/

[32] Anthropic. (2023). "Constitutional AI: Harmlessness from AI Feedback." arXiv preprint arXiv:2212.08073.

[33] Chen, S., et al. (2024). "StruQ: Defending Against Prompt Injection with Structured Queries." arXiv preprint arXiv:2402.06363.

[34] Gu, T., et al. (2019). "BadNets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain." *IEEE Access*, 7.

[35] Schwarzschild, A., et al. (2021). "Just How Toxic is Data Poisoning? A Unified Benchmark for Backdoor and Data Poisoning Attacks." *ICML*.

[36] Ladisa, P., et al. (2023). "SoK: Taxonomy of Attacks on Open-Source Software Supply Chains." *IEEE Symposium on Security and Privacy*.

[37] European Commission. (2024). *EU Artificial Intelligence Act*. Available at: https://artificialintelligenceact.eu/

[38] ISO/IEC JTC 1/SC 42. (2023). *ISO/IEC 42001:2023 - Information technology — Artificial intelligence — Management system*. International Organization for Standardization.

[39] U.S. Department of Health and Human Services. (2024). "HIPAA Compliance for AI/ML Systems in Healthcare." Available at: https://www.hhs.gov/hipaa/

[40] Debenedetti, E., et al. (2024). "AgentDojo: A Framework for Benchmarking LLM Agents Against Adversarial Attacks." arXiv preprint arXiv:2406.13352.

[41] Zhou, S., et al. (2023). "WebArena: A Realistic Web Environment for Building Autonomous Agents." arXiv preprint arXiv:2307.13854.

[42] Zhang, Y., et al. (2024). "Attacking Vision-Language Models with Adversarial Images." *CVPR*.

[43] Kumar, S., et al. (2024). "Audio Adversarial Examples for Speech Recognition Systems." *ICASSP*.

[44] Zou, A., et al. (2024). "PoisonedRAG: Knowledge Poisoning Attacks Against Retrieval-Augmented Generation." arXiv preprint arXiv:2402.07867.

[45] Carlini, N., et al. (2024). "Poisoning Web-Scale Training Datasets is Practical." arXiv preprint arXiv:2302.10149.

[46] WebAssembly Community Group. (2024). *WebAssembly Security Model*. Available at: https://webassembly.org/docs/security/

[47] Google. (2024). *gVisor: Container Runtime Sandbox*. Available at: https://gvisor.dev/

[48] SLSA Framework. (2024). *Supply-chain Levels for Software Artifacts*. Available at: https://slsa.dev/

[49] Kumar, R., et al. (2024). "Transparent ML: Ensuring Model Provenance Through Cryptographic Verification." arXiv preprint arXiv:2403.12847.

[50] NIST. (2020). *Zero Trust Architecture*. NIST Special Publication 800-207. Available at: https://www.nist.gov/publications/zero-trust-architecture

[51] Perez, E., et al. (2022). "Red Teaming Language Models with Language Models." *EMNLP*.

[52] Mehrotra, A., et al. (2023). "Tree of Attacks: Jailbreaking Black-Box LLMs Automatically." arXiv preprint arXiv:2312.02119.

[53] Willison, S. (2023). "Prompt Injection: What's the Worst That Could Happen?" Available at: https://simonwillison.net/2023/Apr/14/worst-that-could-happen/

[54] Zeng, Y., et al. (2024). "How Strong Are LLM-Generated Adversarial Examples? A Study of Adaptive Attacks." arXiv preprint arXiv:2404.08487.

[55] Carlini, N., et al. (2021). "Extracting Training Data from Large Language Models." *USENIX Security Symposium*.

[56] Kolter, J. Z., & Wong, E. (2018). "Provable Defenses Against Adversarial Examples via the Convex Outer Adversarial Polytope." *ICML*.

[57] Tramèr, F., et al. (2020). "Stealing Machine Learning Models via Prediction APIs." *USENIX Security Symposium*.

[58] Song, D., et al. (2024). "AI Security Research at UC Berkeley." Available at: https://people.eecs.berkeley.edu/~dawnsong/

[59] Palo Alto Networks. (2024). "Precision AI: Defending Against Advanced Threats." Available at: https://www.paloaltonetworks.com/cyberpedia/ai-infrastructure-security

[60] CrowdStrike. (2025). "Falcon AI-SPM: Protecting LLM Deployments at Scale." Available at: https://www.crowdstrike.com/en-us/blog/stop-ai-powered-adversaries-fight-fire-with-fire/

[61] Lakera. (2024). "Lakera Guard: Production-Grade Prompt Security." Available at: https://www.lakera.ai/

[62] HiddenLayer. (2023). "Prompt Injection Attacks on LLMs." Available at: https://hiddenlayer.com/innovation-hub/prompt-injection-attacks-on-llms/

[63] Liang, W., et al. (2024). "Adversarial Attacks on Large Language Models Using Regularized Relaxation." arXiv preprint arXiv:2410.19160.

[64] Weng, L. (2023). "Adversarial Attacks on LLMs." *Lil'Log*. Available at: https://lilianweng.github.io/posts/2023-10-25-adv-attack-llm/

[65] Microsoft. (2025). “Protecting Against Indirect Prompt Injection Attacks in MCP.” *Microsoft Developer Blog*. Available at: https://developer.microsoft.com/blog/protecting-against-indirect-injection-attacks-mcp

[66] NSFOCUS. (2024). “The Invisible Battlefield Behind LLM Security Crisis.” Available at: https://nsfocusglobal.com/the-invisible-battlefield-behind-llm-security-crisis/

[67] Nightfall AI. (2024). “MITRE ATLAS: The Essential Guide.” Available at: https://www.nightfall.ai/ai-security-101/mitre-atlas

[68] Practical DevSecOps. (2025). “MITRE ATLAS Framework 2025 - Guide to Securing AI Systems.” Available at: https://www.practical-devsecops.com/mitre-atlas-framework-guide-securing-ai-systems/

[69] Wiz. (2024). “The Threat of Adversarial AI.” Available at: https://www.wiz.io/academy/adversarial-ai-machine-learning

[70] Mukherjee, A. (2024). “Beyond ISO 42001: The Role of ISO/IEC 23894 in AI Risk Management.” *Medium*. Available at: https://medium.com/@mukherjee.amitav/beyond-iso-42001-the-role-of-iso-iec-23894-in-ai-risk-management-7c4f3036544f

[71] AWS. (2024). “AI Lifecycle Risk Management: ISO/IEC 42001:2023 for AI Governance.” Available at: https://aws.amazon.com/blogs/security/ai-lifecycle-risk-management-iso-iec-420012023-for-ai-governance/

[72] Secureframe. (2024). “How to Achieve EU AI Act Compliance and Build Trustworthy AI.” Available at: https://secureframe.com/blog/eu-ai-act-compliance

[73] CISA. (2025). “AI Data Security Guidance.” Available at: https://www.cisa.gov/ai-security-guidance

[74] Alston & Bird. (2025). “NSA, CISA, FBI, and International Partners Issue Joint Guidance on AI Data Security.” Available at: https://www.alston.com/en/insights/publications/2025/06/joint-guidance-ai-data-security

[75] Snyk. (2025). “Securing the Software Supply Chain with AI.” Available at: https://snyk.io/articles/secure-software-supply-chain-ai/

[76] ScienceDirect. (2025). “A Review of Backdoor Attacks and Defenses in Code Large Language Models.” Available at: https://www.sciencedirect.com/science/article/abs/pii/S0950584925000461

[77] Springer. (2025). “When LLMs Meet Cybersecurity: A Systematic Literature Review.” *Cybersecurity*, 8:1. DOI: 10.1186/s42400-025-00361-w.

[78] TechTarget. (2023). “GitHub Copilot Replicating Vulnerabilities, Insecure Code.” Available at: https://www.techtarget.com/searchsecurity/news/366571117/GitHub-Copilot-replicating-vulnerabilities-insecure-code

[79] Petri. (2025). “ChatGPT Flaws Could Let Hackers Steal Data and Hijack Chats.” Available at: https://petri.com/chatgpt-vulnerabilities-hackers-data-theft/

[80] The Hacker News. (2025). "Researchers Find ChatGPT Vulnerabilities That Let Attackers Trick AI Into Leaking Data." Available at:
https://thehackernews.com/2025/11/researchers-find-chatgpt.html

[81] The Register. (2024). "GPT-4 Can Exploit Real Vulnerabilities by Reading Advisories." Available at:
https://www.theregister.com/2024/04/17/gpt4_can_exploit_real_vulnerabilities

[82] MIT News. (2025). "3 Questions: Modeling Adversarial Intelligence to Exploit AI's Security Vulnerabilities." Available at: https://news.mit.edu/2025/3-questions-una-may-o-reilly-modeling-adversarial-intelligence-0129

[83] VentureBeat. (2025). "CrowdStrike, Nvidia Embed Real-Time LLM Defense." Available at: https://venturebeat.com/security/crowdstrike-falcon-now-powers-runtime-defense-in-nvidias-llms

[84] TechMagic. (2024). "HIPAA Compliance AI: Guide to Using LLMs Safely in Healthcare." Available at: https://www.techmagic.co/blog/hipaa-compliant-llms

[85] Wikipedia. (2024). "Prompt Injection." Available at:
https://en.wikipedia.org/wiki/Prompt_injection

[86] Keysight Technologies. (2025). "Prompt Injection Techniques: Jailbreaking Large Language Models via FlipAttack." Available at:
https://www.keysight.com/blogs/en/tech/nwvs/2025/05/20/prompt-injection-techniques-jailbreaking-large-language-models-via-flipattack

[87] Kim, H., et al. (2024). "GPT-4 Jailbreaks Itself with Near-Perfect Success Using Self-Explanation." arXiv preprint arXiv:2405.13077v2.

[88] Yi, J., et al. (2025). "Benchmarking and Defending Against Indirect Prompt Injection Attacks on Large Language Models." *Proceedings of ACM SIGKDD Conference*, Toronto, ON, Canada.

[89] Patterson, D., et al. (2025). "Red Teaming the Mind of the Machine: A Systematic Evaluation of Prompt Injection and Jailbreak Vulnerabilities in LLMs." arXiv preprint arXiv:2505.04806v1.

[90] GitHub Repository. (2025). "Silent Alarm Detector: Behavioral Monitoring for LLM-Generated Code." Claude Code Hooks Security Research Project. Available at:
https://github.com/hah23255/silent-alarm-detector

[91] The Hacker News. (2025). "Malicious ML Models on Hugging Face Leverage Broken Pickle Format to Evade Detection." Available at:
https://thehackernews.com/2025/02/malicious-ml-models-found-on-hugging.html

[92] JFrog Research. (2024). "Over 100 Malicious AI/ML Models Found on Hugging Face Platform." Available at: https://jfrog.com/blog/data-scientists-targeted-by-malicious-hugging-face-ml-models-with-silent-backdoor/

[93] ReversingLabs. (2025). "The Race to Secure the AI/ML Supply Chain." *2025 Software Supply Chain Security Report*. Available at:

https://www.reversinglabs.com/blog/the-race-to-secure-the-aiml-supply-chain-is-on-get-out-front

[94] arXiv. (2025). "The Art of Hide and Seek: Making Pickle-Based Model Supply Chain Poisoning Stealthy Again." arXiv preprint arXiv:2508.19774v1.

[95] Rapid7. (2025). "From .pth to p0wned: Abuse of Pickle Files in AI Model Supply Chains." Available at: https://www.rapid7.com/blog/post/from-pth-to-p0wned-abuse-of-pickle-files-in-ai-model-supply-chains/

Appendix A: Vulnerability Classification Framework

CLI-Specific Attack Vectors:

- A1: Argument Injection
- A2: Environment Variable Exploitation
- A3: Command Substitution
- A4: Configuration File Tampering
- A5: Workspace Trust Bypass

LLM-Specific Attack Vectors:

- L1: Direct Prompt Injection (Jailbreaking)
- L2: Indirect Prompt Injection
- L3: System Prompt Extraction
- L4: Model Extraction
- L5: Data Poisoning
- L6: Backdoor Injection
- L7: Multi-Modal Attacks

Supply Chain Attack Vectors:

- S1: Dataset Poisoning
- S2: Model Checkpoint Tampering
- S3: Malicious Dependencies
- S4: LoRA/PEFT Corruption
- S5: Registry Compromise

Appendix B: Defense Mechanism Taxonomy

Architectural Defenses:

- D1: Containerization (Docker, gVisor)
- D2: WebAssembly Sandboxing
- D3: Least Privilege Execution
- D4: Network Isolation

Input Validation:

- D5: Pattern Detection
- D6: Anomaly Detection
- D7: Delimiter Monitoring
- D8: Structured Queries (Parameterization)

Output Controls:

- D9: Command Execution Filtering
- D10: Shell Escaping
- D11: Safe API Usage
- D12: Human-in-the-Loop Verification

Supply Chain Security:

- D13: Model Signing
- D14: Hash Verification
- D15: SBOM Generation
- D16: Dependency Scanning

Monitoring & Detection:

- D17: Audit Logging
- D18: Provenance Tracking
- D19: Behavioral Analysis
- D20: Anomaly Detection

End of Document

For correspondence regarding this research synthesis, please contact the authors through standard academic channels or via GitHub repository: https://github.com/hah23255/claude-hooks-security-research

This document is formatted for submission to arXiv.org (cs.CR, cs.AI, cs.SE) and distribution as industry white paper. Version 1.1 | November 04, 2025