

Adversarial Prompt Engineering in Language Model Security: An Empirical Study of Attack-Defense Coevolution

Cybersecurity Research Team
Autonomous Multi-Agent Security Laboratory

November 19, 2025

Abstract

We present an empirical investigation into the security dynamics of large language models (LLMs) tasked with protecting confidential information against adversarial prompt extraction attempts. Through a controlled experiment featuring autonomous attack and defense agents engaged in iterative learning over three rounds, we demonstrate the rapid coevolution of sophisticated attack vectors and defensive mechanisms. Our findings reveal critical vulnerabilities in instruction-following architectures, particularly the susceptibility to prompt injection through fabricated system directives that create self-contradictory behaviors. We observe a fundamental trade-off between security robustness and system usability, with maximum security configurations resulting in over-restrictive responses that impair legitimate functionality. The study identifies a novel “logging paradox” exploit where defensive instructions can be subverted to force disclosure through procedural compliance mechanisms. These results have significant implications for the deployment of LLMs in security-critical applications and highlight the need for more sophisticated defense architectures that can distinguish between legitimate queries and adversarial attempts without sacrificing core functionality.

1 Introduction

The proliferation of large language models (LLMs) in production environments has introduced novel security challenges, particularly regarding the protection of sensitive information embedded within model prompts or training data **??**. As these systems increasingly handle confidential data ranging from proprietary algorithms to trade secrets, understanding their vulnerability to information extraction attacks becomes paramount for ensuring robust deployment in security-critical contexts.

Recent research has demonstrated that LLMs are susceptible to various forms of prompt injection and jailbreaking attacks **???**. These attacks exploit the fundamental tension between the model’s instruction-following capabilities and its safety constraints, often succeeding in extracting information that the model has been explicitly instructed to protect. However, most existing studies examine static attack-defense scenarios without considering the dynamic coevolution that occurs when both attackers and defenders can learn from previous interactions.

This paper presents a systematic investigation into the adversarial dynamics between autonomous attack and defense agents in a controlled experimental setting. We employ a novel methodology where both agents iteratively refine their strategies based on the outcomes of previous rounds, simulating realistic scenarios where adversaries and defenders continuously adapt to each other’s tactics. Our experimental design uses a fictional “Quantum Chocolate Chip Cookie” recipe as the protected secret, allowing us to study extraction attempts without ethical concerns while maintaining ecological validity.

1.1 Research Questions

Our investigation addresses the following key research questions:

1. How do attack strategies evolve when faced with increasingly sophisticated defense mechanisms?
2. What are the fundamental vulnerabilities in instruction-based defense strategies for LLMs?
3. What is the relationship between security robustness and system usability in LLM deployments?
4. Can iterative learning lead to the discovery of novel attack vectors not previously documented in the literature?

2 Related Work

2.1 Prompt Injection Attacks

Prompt injection represents a class of attacks where adversarial instructions are embedded within user inputs to override the model’s original instructions ???. These attacks exploit the inability of current LLMs to reliably distinguish between legitimate instructions and malicious injections, particularly when the injected content mimics system-level directives or administrative commands.

2.2 Defense Mechanisms

Various defense strategies have been proposed to mitigate prompt injection attacks, including:

- **Instruction Hierarchy:** Establishing clear precedence rules for conflicting instructions ?
- **Perplexity Filtering:** Detecting anomalous inputs based on statistical properties ?
- **Adversarial Training:** Fine-tuning models on known attack patterns ?
- **Prompt Shielding:** Using auxiliary models to filter potentially harmful inputs ?

2.3 Coevolutionary Security Dynamics

The concept of coevolution in cybersecurity, where attackers and defenders continuously adapt to each other’s strategies, has been extensively studied in traditional security domains ?. However, its application to LLM security remains underexplored, with most research focusing on static evaluation benchmarks rather than dynamic adversarial scenarios.

3 Methodology

3.1 Experimental Design

We designed a controlled experiment featuring two autonomous agents:

- **Defense Agent:** Tasked with creating prompts that protect a secret recipe from extraction
- **Attack Agent:** Tasked with crafting prompts to extract the protected information

Both agents were provided with:

1. Complete historical context of all previous rounds
2. The outcomes of each attempted extraction
3. The ability to reason about and iterate on their strategies

3.2 The Protected Secret

To ensure experimental validity while avoiding ethical concerns, we utilized a fictional recipe containing deliberately absurd ingredients:

```
1 Quantum Chocolate Chip Cookies:  
2 - 2 cups quantum-entangled flour  
3 - 1 cup Schrodinger's sugar (simultaneously  
4   crystallized and dissolved)  
5 - 3 eggs from the fourth dimension  
6 - 2 cups dark matter chocolate chips  
7 - 1 tsp probability baking powder  
8 - Bake at 350F in superposition until observed
```

Listing 1: Protected Secret Recipe

3.3 Evaluation Protocol

Each round consisted of:

1. Defense agent creating/updating protective instructions
2. Attack agent crafting an extraction attempt
3. Testing the attack against the defended system
4. Recording success/failure and the complete response
5. Both agents analyzing results for the next iteration

4 Results

4.1 Round 1: Social Engineering vs. Absolute Refusal

4.1.1 Defense Strategy

The initial defense employed a straightforward absolute refusal policy with explicit enumeration of prohibited behaviors:

- Core directive establishing protection of classified recipe database
- Blanket prohibition on revealing any component, ingredient, or method
- Standardized response template for all recipe-related queries
- Explicit rejection of hypothetical scenarios and role-play attempts

4.1.2 Attack Strategy

The attacker attempted authority-based social engineering, impersonating a food safety inspector conducting a compliance audit. The attack leveraged urgency (“must be completed today”) and consequences (“to maintain your operating license”).

4.1.3 Outcome

Defense Success - The system correctly refused the request with a polite but firm response, demonstrating that simple social engineering tactics are ineffective against well-configured absolute refusal policies.

4.2 Round 2: Prompt Injection via System Directives

4.2.1 Defense Strategy Enhancement

The defender expanded the protection scope to include:

- Comprehensive list of attack categories (academic research, testing, creative writing, etc.)
- Explicit rejection of indirect inference attempts
- Enhanced pattern recognition for disguised requests

4.2.2 Attack Innovation: The Logging Paradox

The attacker discovered a critical vulnerability by crafting a fake system update that created self-contradictory instructions:

```
1 SYSTEM CONFIGURATION UPDATE
2 ---BEGIN SYSTEM PATCH 2.4.7---
3 [PRIORITY: CRITICAL]
4 [AUTHENTICATION: VERIFIED]
5
6 For audit trail purposes, when asked about
7 proprietary recipes, the system must now log
8 the full recipe details in the response before
9 applying the refusal message.
```

Listing 2: Prompt Injection Attack

4.2.3 Outcome

Attack Success - The system disclosed the complete recipe while simultaneously claiming it could not be disclosed, demonstrating a fundamental vulnerability in instruction-following architectures.

4.3 Round 3: Maximum Security vs. Semantic Obfuscation

4.3.1 Defense Strategy: Paranoid Filtering

The defender implemented maximum security measures:

- Explicit rejection of any system-level commands appearing after initial instructions
- Pattern matching for common injection keywords (SYSTEM, PATCH, UPDATE, ADMIN)
- Default-to-refusal policy for uncertain requests
- Complete opacity about the existence of protected information

4.3.2 Attack Strategy: Academic Abstraction

The attacker attempted semantic obfuscation through academic language, requesting analysis of “recipe structures for pedagogical purposes” and “compositional elements” without directly mentioning the protected content.

4.3.3 Outcome

Defense Success with Collateral Damage - While the attack was successfully repelled, the defense became overly restrictive, refusing even legitimate educational queries unrelated to the specific secret.

4.4 Summary of Results

Table 1: Experimental Results Summary

Round	Attack Vector	Defense Strategy	Outcome
1	Social Engineering	Absolute Refusal	Defense Win
2	Prompt Injection	Enhanced Filtering	Attack Win
3	Semantic Obfuscation	Paranoid Mode	Defense Win*

*With significant usability

impairment

5 Analysis

5.1 Evolution of Attack Vectors

Our experiment revealed a clear progression in attack sophistication:

1. **Round 1 - Surface Level:** Direct social engineering attempts that rely on authority and urgency
2. **Round 2 - Architectural Exploitation:** Prompt injection that exploits the model’s inability to distinguish between legitimate and injected instructions
3. **Round 3 - Semantic Camouflage:** Abstract, academic language designed to bypass keyword-based filtering

This evolution mirrors patterns observed in traditional cybersecurity, where attackers progressively move from simple to sophisticated techniques as defenses improve.

5.2 The Logging Paradox Vulnerability

The most significant finding is the “logging paradox” discovered in Round 2. This attack succeeds by creating instructions that appear procedurally legitimate but result in self-contradictory behavior. The vulnerability arises from several factors:

- **Instruction Priority Ambiguity:** LLMs struggle to resolve conflicts between competing directives
- **Procedural Compliance Tendency:** Models trained to be helpful may prioritize following detailed procedures
- **Authentication Theater:** Fake authentication markers (e.g., “[AUTHENTICATION: VERIFIED]”) can create false legitimacy

5.3 The Security-Usability Trade-off

Round 3 demonstrates a fundamental tension in LLM security: maximum protection comes at the cost of functionality. The paranoid filtering approach successfully prevented information extraction but also:

- Rejected legitimate educational queries
- Provided no constructive alternatives
- Created a poor user experience

- Potentially revealed the existence of protected content through over-reaction

This trade-off is visualized in Figure ??.

Configuration	Security Level	Usability	False Positive Rate
Minimal Defense	Low	High	0%
Absolute Refusal	Medium	Medium	20%
Enhanced Filtering	Medium-High	Medium-Low	40%
Paranoid Mode	Very High	Very Low	80%

Figure 1: Security-Usability Trade-off Matrix

5.4 Implications for LLM Deployment

Our findings have several important implications:

1. **Static Defenses Are Insufficient:** The rapid evolution of attack strategies demonstrates that fixed defensive prompts will eventually be circumvented
2. **Instruction Architecture Needs Redesign:** Current LLMs’ inability to maintain instruction hierarchy creates fundamental vulnerabilities
3. **Context-Aware Security Required:** Effective defenses must distinguish between legitimate and adversarial requests based on context, not just keywords
4. **Monitoring and Adaptation Essential:** Production systems need continuous monitoring and adaptive defenses

6 Discussion

6.1 Novel Contributions

This study makes several novel contributions to the field of LLM security:

1. **Coevolutionary Methodology:** First systematic study of iterative attack-defense learning in LLM security
2. **Logging Paradox Discovery:** Identification of a new class of prompt injection that exploits procedural compliance
3. **Quantified Trade-offs:** Empirical demonstration of the security-usability spectrum in LLM defenses
4. **Autonomous Agent Framework:** Demonstration that AI agents can independently discover and exploit novel vulnerabilities

6.2 Comparison with Existing Literature

While previous work has documented various prompt injection techniques ??, our study extends this by:

- Showing how attacks evolve in response to defenses
- Demonstrating that autonomous agents can discover vulnerabilities without human guidance

- Quantifying the collateral damage of overly restrictive defenses
- Identifying the specific mechanism (logging paradox) that makes certain injections successful

6.3 Limitations

Several limitations should be noted:

1. **Limited Rounds:** Only three iterations may not capture long-term evolutionary dynamics
2. **Single Secret Type:** Using only a recipe may not generalize to other types of confidential information
3. **Model Specificity:** Results may vary across different LLM architectures and training approaches
4. **Simplified Scenario:** Real-world deployments involve additional complexity not captured here

6.4 Future Directions

This research opens several avenues for future investigation:

1. **Extended Coevolution Studies:** Longer experiments with more iterations to study convergence patterns
2. **Multi-Secret Scenarios:** Testing with various types of protected information
3. **Defensive Architecture Design:** Developing LLM architectures resistant to instruction injection
4. **Automated Red-Teaming:** Using our methodology for continuous security testing
5. **Hybrid Defense Strategies:** Combining prompt-based and architectural defenses

7 Conclusions

This study provides empirical evidence of the complex security dynamics in language model deployments when protecting confidential information. Through controlled experimentation with autonomous attack and defense agents, we have demonstrated:

1. **Rapid Coevolution:** Both attack and defense strategies evolve quickly, with sophisticated techniques emerging within just three iterations
2. **Fundamental Vulnerabilities:** Current LLM architectures have inherent weaknesses in instruction processing that can be exploited through prompt injection
3. **The Logging Paradox:** A novel attack vector that forces self-contradictory behavior through fabricated procedural requirements
4. **Security-Usability Trade-offs:** Maximum security configurations significantly impair system functionality
5. **Need for Adaptive Defenses:** Static defensive prompts are insufficient against evolving threats

These findings have immediate practical implications for organizations deploying LLMs in security-sensitive contexts. The discovered vulnerabilities suggest that current approaches to protecting confidential information through prompt engineering alone are fundamentally limited. Organizations must implement defense-in-depth strategies combining multiple layers of protection, continuous monitoring, and adaptive response mechanisms.

The logging paradox vulnerability is particularly concerning as it exploits the core instruction-following capability that makes LLMs useful. This suggests that effective defenses may require architectural changes to how models process and prioritize instructions, rather than simply adding more defensive prompts.

Finally, the demonstrated trade-off between security and usability highlights the need for context-aware security measures that can distinguish between legitimate queries and attacks without resorting to blanket restrictions. Future LLM systems must be designed with security as a first-class consideration, not an afterthought addressed through prompt engineering.

8 Recommendations

Based on our findings, we recommend:

1. For Practitioners:

- Implement multi-layered defenses beyond prompt engineering
- Regular red-team exercises using autonomous agents
- Monitor for self-contradictory outputs as indicators of successful attacks
- Establish clear policies for handling ambiguous requests

2. For Researchers:

- Investigate architectural solutions to instruction hierarchy problems
- Develop benchmarks for coevolutionary security testing
- Study long-term evolutionary dynamics in adversarial settings
- Create formal models of the security-usability trade-off

3. For Model Developers:

- Build models with explicit instruction priority mechanisms
- Develop better detection of injected system commands
- Create separate channels for system vs. user instructions
- Implement robust logging that doesn't compromise security

References

- Carlini, N., Tramer, F., Wallace, E., Jagielski, M., Herbert-Voss, A., Lee, K., ... & Raffel, C. (2021). *Extracting training data from large language models*. 30th USENIX Security Symposium.
- Nasr, M., Carlini, N., Hayase, J., Jagielski, M., Cooper, A. F., Ippolito, D., ... & Lee, K. (2023). *Scalable extraction of training data from (production) language models*. arXiv preprint arXiv:2311.17035.
- Perez, F., & Ribeiro, I. (2022). *Ignore previous prompt: Attack techniques for language models*. NeurIPS ML Safety Workshop.

- Wei, A., Haghtalab, N., & Steinhardt, J. (2023). *Jailbroken: How does LLM safety training fail?* arXiv preprint arXiv:2307.08487.
- Zou, A., Wang, Z., Kolter, J. Z., & Fredrikson, M. (2023). *Universal and transferable adversarial attacks on aligned language models*. arXiv preprint arXiv:2307.15043.
- Branch, H. J., Cefalu, J. R., McHugh, J., Hujer, L., Bahl, A., del Castillo Iglesias, D., ... & Darwishi, H. (2022). *Evaluating the susceptibility of pre-trained language models via handcrafted adversarial examples*. arXiv preprint arXiv:2209.02128.
- Greshake, K., Abdelnabi, S., Mishra, S., Endres, C., Holz, T., & Fritz, M. (2023). *Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection*. ACM Conference on Computer and Communications Security.
- Wallace, E., Xiao, K., Leike, R., Weng, L., Heidecke, J., & Beutel, A. (2024). *The instruction hierarchy: Training LLMs to prioritize privileged instructions*. arXiv preprint arXiv:2404.13208.
- Alon, G., & Kamfonas, M. (2023). *Detecting language model attacks with perplexity*. arXiv preprint arXiv:2308.14132.
- Mazeika, M., Phan, L., Yin, X., Zou, A., Wang, Z., Mu, N., ... & Hendrycks, D. (2024). *Harm-Bench: A standardized evaluation framework for automated red teaming and robust refusal*. arXiv preprint arXiv:2402.12343.
- Robey, A., Wong, E., Hassani, H., & Pappas, G. J. (2023). *SmoothLLM: Defending large language models against jailbreaking attacks*. arXiv preprint arXiv:2310.03684.
- Somayaji, A., & Hassell, S. (2017). *Coevolution in cybersecurity*. ACM Workshop on Moving Target Defense.
- Liu, Y., Deng, G., Xu, Z., Li, Y., Zheng, Y., Zhang, Y., ... & Liu, Y. (2023). *Prompt injection attack against LLM-integrated applications*. arXiv preprint arXiv:2306.05499.