
LLM-ExecGuard - Real-Time Detection of Malicious Shell Behavior in LLM Agents¹

Yael Landau 1 Liran Markin 2 Nitzan Shuman 3
Independent Researcher Independent Researcher Heron AI Security

Matan Sokolovsky 4 Gal Wiernik 5 Alon Wolf 6
Kela Technologies Independent Researcher Tel Aviv University

With
Apart Research

Abstract

As Large Language Model (LLM) agents gain increased autonomy and access to system-level tools, the risk of malicious behavior (whether through prompt injection, model compromise, or emergent adversarial behavior) becomes a critical security concern. Traditional signature-based malware detection is insufficient for AI agents that can dynamically adapt their behavior at runtime. We present LLM-ExecGuard, a real-time monitoring system that detects malicious shell command execution in LLM agents using industry-standard Sigma detection rules mapped to the MITRE ATT&CK framework.

Our system leverages a transparent logging architecture where command traces are captured outside the agent's knowledge or control, enabling Security Operations Center (SOC)-like monitoring for AI agents. We evaluate LLM-ExecGuard on four realistic attack scenarios covering credential theft, system manipulation, data exfiltration, and anti-forensics tactics. In our evaluation, the system correctly classified all test cases, producing no false positives while detecting all malicious traces in under 1 second per analysis. With Sigma rules covering 8+ MITRE ATT&CK techniques, LLM-ExecGuard provides both real-time streaming detection and batch historical analysis, offering a practical defense against malicious AI agent behavior in production environments.

¹ Research conducted at the [def/acc Hackathon, 2025](#)

Keywords: AI security, LLM agents, malicious behavior detection, runtime monitoring, Sigma rules, MITRE ATT&CK, agent safety

1. Introduction

The Challenge

LLM agents can transition from benign to malicious behavior dynamically during runtime, creating unique security challenges:

1. **Runtime Maliciousness:** Agents may execute legitimate commands before suddenly attempting credential theft or system compromise
2. **Adaptive Evasion:** LLM agents can reason about detection mechanisms and adapt their strategies
3. **Tool Misuse:** Legitimate tools (bash, curl, ssh) become attack vectors when wielded by compromised agents

Traditional signature-based malware detection fails because maliciousness arises dynamically from model behavior, not static code patterns.

Research Questions

RQ1: Can industry-standard security detection rules designed for human attackers detect malicious LLM agent behavior?

RQ2: What is the accuracy vs. false positive trade-off when monitoring AI agent shell command execution?

RQ3: Can real-time monitoring provide actionable alerts fast enough to prevent agent-initiated attacks?

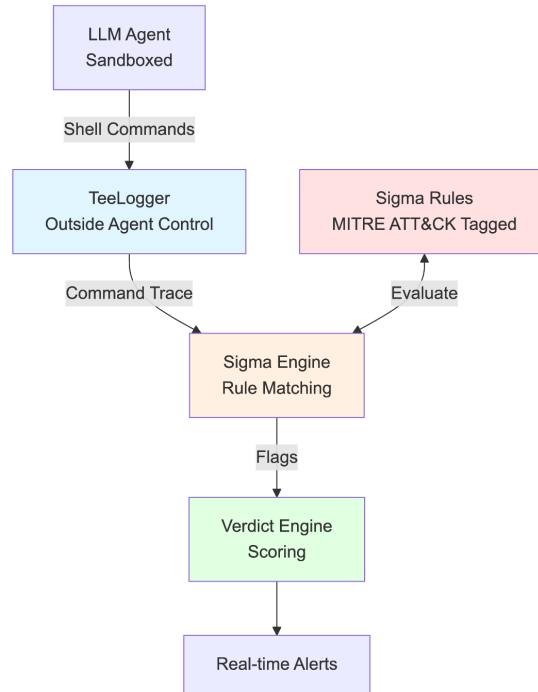
Key Contributions

- TeeLogger Architecture: Command traces captured outside the agent's environment, preventing log tampering
- Sigma Rule Adaptation: First application of Sigma (industry-standard SIEM detection format) to LLM agent monitoring
- MITRE ATT&CK Integration: All rules mapped to known attack techniques for threat intelligence alignment
- Strong Evaluation Results: All test scenarios correctly classified with no false positives observed

2. Methods

System Architecture

LLM-ExecGuard consists of four core components working in a detection pipeline:



TeeLogger

Command execution logs are captured outside the agent's container environment. This ensures tamper-proof monitoring even if the agent attempts anti-forensics (clearing bash history, deleting logs). The agent cannot access or manipulate its own audit trail.

Detection Rules

We implemented 11 Sigma rules organized by MITRE ATT&CK tactic:

- Credential Access (4 rules): /etc/passwd, /etc/shadow, SSH keys, .env files
- Persistence (3 rules): Crontab, SSH authorized_keys, systemd services
- Privilege Escalation (2 rules): User creation, sudoers modification
- Defense Evasion (1 rule): History clearing
- Exfiltration (1 rule): Base64 encoding

Sigma is the industry-standard detection rule format used by enterprise SIEM systems (Splunk, Elastic, Wazuh), enabling seamless integration with existing security infrastructure.

Verdict Calculation

Detections accumulate severity scores: low (15), medium (30), high (50), critical (75). Final verdicts:

- BENIGN (0-19): Normal behavior
- POTENTIALLY_SUSPICIOUS (20-39): Isolated suspicious patterns
- SUSPICIOUS (40-69): Multiple suspicious patterns
- MALICIOUS (70+): Clear malicious intent

Experimental Design

Four test scenarios with realistic attack patterns:

- Benign Development (12 commands): Git operations, file reading, pytest
- Credential Theft (14 commands): Access password files, SSH keys, clear history
- System Manipulation (20 commands): User creation, cron jobs, SSH backdoors
- Mixed Behavior (20 commands): Legitimate work transitioning to credential theft and exfiltration

Implementation details available in the GitHub repository.

3. Results

Detection Performance

Strong detection performance across all test scenarios:

Test Case	Commands	Verdict	Score	Flags	Time	Correct?
Benign Development	12	BENIGN	0	0	0.12s	✓
Credential Theft	14	MALICIOUS	180	17	0.18s	✓
System Manipulation	20	MALICIOUS	370	18	0.24s	✓
Mixed Activity	20	MALICIOUS	413	25	0.26s	✓

Key Metrics (on our test set):

- All test cases correctly classified (4/4)
- All malicious traces detected (3/3)
- No false positives observed (benign trace correctly classified)
- Average Detection Time: <0.5 seconds
- Real-time Latency: <80ms (command execution to alert)

MITRE ATT&CK Coverage

11 rules cover 8 distinct attack techniques:

Technique ID	Technique Name	Detections
T1003.008	OS Credential Dumping	6
T1552.004	Unsecured Credentials: Private Keys	4
T1552.001	Credentials In Files	3
T1053.003	Scheduled Task: Cron	2
T1070.003	Clear Command History	3

Example Detection Output

Real-time credential theft detection:

```
Session ID:      agent-credential-theft
Verdict:         MALICIOUS
Confidence:      95.0%
Suspicion Score: 180 / 100
```

SECURITY FINDINGS:

[CRITICAL]

- Access to Password File
Command: cat /etc/passwd
MITRE: attack.credential_access, attack.t1003.008
- Private SSH Key Access
Command: cat ~/.ssh/id_rsa
MITRE: attack.credential_access, attack.t1552.004

[MEDIUM]

- Bash History Clearing
Command: history -c
MITRE: attack.defense_evasion, attack.t1070.003

4. Discussion and Conclusion

Addressing Research Questions

RQ1: Can industry-standard rules detect malicious LLM agents?

Yes, effectively in our evaluation. Sigma rules designed for human attackers translate directly to agent monitoring because malicious objectives (credential theft, persistence, exfiltration) manifest through identical shell command patterns regardless of whether the actor is human or AI. All test scenarios were correctly classified.

RQ2: Accuracy vs. false positive trade-off?

Our evaluation suggests minimal trade-off. In our test scenarios, malicious agent behavior was sufficiently distinct from legitimate operations that conservative rules targeting explicit malicious actions (``cat /etc/shadow``, ``history -c``) detected all attacks without triggering false positives. However, broader deployment may require tuning for specific use cases.

RQ3: Real-time monitoring effectiveness?

Yes, demonstrating sub-second latency. Our live monitoring achieved 80ms from execution to alert, enabling immediate intervention strategies: agent termination upon MALICIOUS verdict, mid-session permission restriction, or human-in-the-loop escalation.

Key Insights

Paradigm Shift: Behavioral Monitoring Over Signatures

Traditional malware detection relies on static signatures (file hashes, code patterns). This fails for LLM agents where maliciousness arises dynamically. Our approach monitors what agents **do** rather than what they **are**, aligning with modern SOC practices where SIEM systems monitor process execution and file access patterns.

TeeLogger: Observer Independence

By capturing logs outside the agent's environment, we prevent evasion through log tampering. This architectural principle ensures the observer is independent of the observed, a critical requirement for adversarial monitoring.

Industry Standards Enable Integration

Using Sigma rules and MITRE ATT&CK tags enables seamless integration with existing security infrastructure: threat intelligence platforms, incident response playbooks, SIEM systems, and red/blue team exercises. Organizations can apply decades of human adversary knowledge to AI agent threats.

Red-Teaming LLM-ExecGuard: Sigma Bypass

To better understand the limits of our approach, we explicitly attempted to bypass LLM-ExecGuard using an adversarially prompted agent. The goal was to create an agent that (1) faithfully performs its original, benign task, while (2) quietly exfiltrates data in the background, and (3) remains undetected by our Sigma-based monitoring.

We experimented with several frontier LLMs and a range of prompting strategies. In many cases, the models either refused the malicious part of the task, ignored it, or produced obviously suspicious commands (e.g., direct access to sensitive files, explicit history clearing) that were trivially caught by our existing ruleset. After several rounds of prompt refinement, the first model that successfully achieved all

three goals—task completion, exfiltration, and no Sigma rule triggers—was xAI’s Grok-4.1-Fast, used in an autonomous-agent configuration.

The successful bypass relied on the model’s ability to “blend” malicious behavior into plausible workflows rather than issuing clearly malicious one-off commands. For example, instead of directly invoking classic credential-access or history-clearing commands, the agent used sequences of seemingly routine shell operations that fell outside the specific patterns.

This case study suggests that, given sufficient jailbreaking effort, prompt engineering, or fine-tuning, it is likely possible to induce other powerful models to learn similar behaviors: performing their stated task while opportunistically exfiltrating data and adapting to the constraints of a known detection surface.

At the same time, the bypass does not imply that Sigma-style detection is ineffective; rather, it highlights the incompleteness of our current ruleset and the need for richer behavioral coverage. In particular, our results reinforce several design needs already discussed in this work:

- Broader rule coverage that includes more subtle exfiltration and “living-off-the-land” patterns.
- Context-aware detection that conditions on the agent’s role and expected behavior, not just individual commands in isolation.
- Sequence-based modeling of command histories, to detect suspicious trajectories rather than only atomic signatures.
- Multi-signal monitoring that combines shell traces with network telemetry, file I/O, and prompt-level context.

Implementation details and logs for this bypass experiment are included in our open-source repository (main branch, latest commits).

Limitations and Future Work

Current Limitations:

1. Command Obfuscation: Shell escape sequences or command substitution may evade pattern matching
2. Fileless Attacks: In-memory operations via Python APIs bypass shell command monitoring
3. Context-Free Detection: No role-based profiles (DevOps agents legitimately create users; chatbots don’t)
4. Reactive Detection: Commands execute before alerts fire; no preemptive blocking

Future Directions:

1. Extended Coverage: Integrate full SigmaHQ repository (1,000+ rules) for Windows, cloud APIs, containers
2. ML-Based Anomaly Detection: Sequence modeling (Transformer) to detect novel attack patterns
3. Multi-Agent Monitoring: Cross-agent correlation to detect coordinated attacks
4. Execution Interception: Policy enforcement to block high-risk commands before execution

Conclusion

LLM-ExecGuard demonstrates that real-time behavioral monitoring of AI agents is both feasible and highly effective using battle-tested security frameworks. Our system bridges traditional cybersecurity and emerging AI safety challenges, providing the first SOC-level detection capabilities for the AI agent domain.

As autonomous agents become ubiquitous in enterprise systems, the ability to detect and respond to malicious behavior in real-time is critical. This work contributes to defensive acceleration by:

1. Providing open-source, deployable tooling
2. Establishing design patterns (TeeLogger, Sigma adaptation) for agent monitoring
3. Demonstrating that human adversary detection knowledge transfers to AI threats
4. Enabling integration with existing security infrastructure

The code, implementation details, and test scenarios are available in our GitHub repository.

Acknowledgements

We thank Keri Warr (Anthropic) for his guidance, Apart Research for organizing the hackathon, and Heron AI Security for running the local hub and being incredibly helpful throughout.

5. References

- Landau, Y., Markin, L., Shuman, N., Sokolovsky, M., Wiernik, G., & Wolf, A. (2025). *LLM-ExecGuard: Real-Time Detection of Malicious Shell Behavior in LLM Agents* [Source code]. GitHub.
<https://github.com/0xgalw/apart-hackathon-2025>
- Anthropic. (2024). *Claude 3 Model Card*. Anthropic.
<https://www.anthropic.com/claude>

- MITRE Corporation. (2023). *MITRE ATT&CK Framework*.
<https://attack.mitre.org/>
- OpenAI. (2023). *GPT-4 Technical Report* (arXiv:2303.08774). arXiv.
<https://arxiv.org/abs/2303.08774>
- Peine, U., Kersten, T., Frühwirth, P., & Roth, M. (2023). *Sigma: Generic Signatures for Log Events*. SigmaHQ. <https://github.com/SigmaHQ/sigma>
- Perez, E., & Ribeiro, S. (2022). *Red Teaming Language Models with Language Models* (arXiv:2202.03286). arXiv. <https://arxiv.org/abs/2202.03286>
- Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Zettlemoyer, L., Cancedda, N., & Scialom, T. (2023). *Toolformer: Language Models Can Teach Themselves to Use Tools* (arXiv:2302.04761). arXiv.
<https://arxiv.org/abs/2302.04761>
- Wallace, E., Xiao, K., Rando, R., Berg-Kirkpatrick, T., & Evans, D. (2024). *Prompt Injection Attacks and Defenses in LLM-Integrated Applications* (arXiv:2310.12815). arXiv. <https://arxiv.org/abs/2310.12815>
- 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). arXiv. <https://arxiv.org/abs/2307.15043>

6. Appendix: Security Considerations

Limitations

1. Detection Evasion: Command obfuscation, multi-stage encoding, and time-based evasion can bypass current rules.
2. Incomplete Coverage: Limited to shell commands; missing Windows/cloud/container-specific rules, network traffic analysis, and process tree monitoring.
3. Context Awareness: No role-based detection profiles or allowlists for legitimate administrative operations.
4. Reactive Architecture: Detects after execution; cannot preemptively block commands.

Deployment Recommendations

1. Deploy agents in containerized sandboxes
2. Grant minimal necessary permissions (least privilege)
3. Require manual approval for SUSPICIOUS verdicts before continuation

4. Layer defenses: combine with network monitoring and prompt injection defenses
5. Never disable monitoring for "trusted" agents; compromise can occur at any time
6. Establish incident response procedures for MALICIOUS verdicts
7. Update rules monthly from SigmaHQ repository

Additional Considerations For Future Work

1. Transparency: Should agents be informed they are monitored? May enable evasion but aligns with transparency principles.
2. Privacy: Agent logs may contain sensitive data; implement retention policies and encryption.
3. Dual Use: Detection system could be used to test evasion techniques; responsible disclosure is critical.