

Securing Generative-AI and AIOps Systems Against Prompt Injection, Data Poisoning, and Adversarial Attacks

Jayasanthosh Ankathi
Independent Researcher
Baltimore, USA

ABSTRACT

Generative AI and AIOps are increasingly embedded in enterprise observability, automated remediation, service management and the software supply chain, which widens the attack surface from prompts to training data and model updates. This work proposes and evaluates a multi-layered defense against prompt injection, data poisoning, adversarial input manipulation and supply-chain tampering. A controlled dataset of 172 attack instances was constructed, with an equal number of instances for each threat class. Every instance was run against an unsecured baseline, an input-gateway configuration and a full defense configuration that combines data screening, provenance checks, adversarial validation, policy enforcement, continuous monitoring and incident response. The experimental workflow was implemented in Python using a structured security test harness, with Matplotlib for visual analysis and logging utilities for reproducible evidence capture. The results show that although isolated controls can stop attacks that are clearly malicious, vulnerabilities remain at cross-layer points and are exposed when attacks exploit trusted context, polluted training material or operational automation. The layered approach further reduces the likelihood of successful attacks and preserves useful model behavior while improving the ability to detect, contain and recover from an attack. The paper therefore recommends a practical security pattern for organizations deploying generative-AI and AIOps components in production environments, where governance, auditability and controlled automation must work together rather than as standalone security considerations.

General Terms

Security, Artificial Intelligence, Machine Learning, Algorithms.

Keywords

Prompt injection, data poisoning, adversarial attacks, generative AI, AIOps.

1. INTRODUCTION

Generative-AI systems are moving beyond experimental assistants into enterprise processes: retrieving knowledge, generating software changes, correlating telemetry, suggesting fixes and orchestrating operational activities. At the same time, AIOps platforms are shifting from alert aggregation to automated reasoning over logs, incidents, infrastructure state and service dependencies. When these capabilities are combined, the resulting environment not only processes untrusted natural-language content but can also influence high-impact operational decisions. This convergence changes the security problem. Conventional application controls assume that instructions, data and executable actions travel in separate channels, whereas recent AI systems turn natural-language

context into action and may treat retrieved text, tool output or monitoring evidence as a reliable guide. Attackers therefore gain opportunities to influence decisions without exploiting a typical memory bug or authentication flaw. Biggio and Roli established a lasting security perspective on learning systems in which adversarial manipulation is possible during both training and deployment [1]. The survey by Akhtar and Mian also showed that small, well-structured inputs can divert the behavior of deep-learning models even when they appear to be normal observations [2]. Yuan et al. systematized attack and defense techniques for deep models and stressed that deployment robustness should be evaluated against intentional manipulation rather than normal test data [3]. These concepts now extend beyond visual classifiers, because language models can receive instructions that mix user intent with adversarial content.

Prompt injection is particularly important because system rules, user requests, retrieved documents, external web pages, tool responses and conversation memory can all enter a single reasoning context. Any lower-trust source may contain malicious instructions that override a higher-priority policy or cause the model to reveal protected information, misuse a tool or suppress an alert. Kwon and Pak demonstrated how text-based injection can be crafted to bypass simple filtering and exploit the tendency of a model to treat seemingly legitimate text as valid [4]. In a related line of security research, Yao et al. showed that large language models are exposed to risks across training, deployment, privacy and application integration [5]. Prompt injection is potentially more serious in an AIOps environment, since the model may do more than respond to text: it may rank incidents, provide instructions, update tickets, summarize root causes or orchestrate actions. A successful injection can therefore turn linguistic manipulation into operational manipulation.

Data poisoning poses a different but complementary threat. Training corpora, fine-tuning sets, feedback records, retrieval indexes, telemetry archives and incident history can all affect the future behavior of a model. If an adversary pollutes these sources, the model may learn false associations or recall malicious information that appears valid because it resides in internal storage. A survey by Wang et al. classifies poisoning as a training-stage threat that can alter learned behavior without the attacker having to manipulate every subsequent input [6]. Experimental studies by Yerlikaya and Bahtiyar showed that poisoning can undermine a range of learning algorithms across different application settings [7]. For generative-AI and AIOps platforms, the same risk extends to model updates and operational knowledge bases, where compromised provenance can silently alter the future recommendations of the model.

Adversarial inputs add a further layer of risk because they target the inference process with crafted content that causes unsafe or inaccurate behavior while still appearing plausible to standard

validation. The review by Xu et al. showed that adversarial methods apply to images, graphs and text, which indicates that the concept is not specific to one data type [8]. A survey by Chakraborty et al. found that defense strategies often fail when they are tested individually against known attacks [9]. From a cybersecurity standpoint, Rosenberg et al. bridged the gap between adversarial machine-learning attacks and practical defensive activities, supporting the view that domain-driven protection, rather than generic accuracy testing, is necessary [10]. Recent cyber-defense studies also indicate that large language models can both aid threat detection and raise new trust and misuse issues; the survey by Chen et al. analyzed this dual role in threat-analysis settings [11]. The survey by Zhou et al. similarly raises concerns about bias, misuse, privacy and adversarial attacks on large language models [12].

2. REVIEW OF LITERATURE

Research on adversarial artificial intelligence has progressed from the study of weak classifiers to entire model ecosystems, in which training data, inference inputs, service integrations and downstream behaviors all create opportunities for manipulation. Earlier studies established that a model may perform well on standard evaluation material yet fail on purposely designed inputs. Biggio and Roli framed this as a lifecycle security problem and distinguished between attacks that affect learning and attacks that affect deployed decisions [1]. The taxonomy of Yuan et al. later unified the methods for generating adversarial examples and defenses, and highlighted the common gap between standard predictive performance and performance in a hostile environment [3]. Akhtar and Mian surveyed adversarial attacks in the computer-vision domain and highlighted the challenge of constructing defenses that resist attacks beyond those seen during training [2]. Although these studies predate widespread enterprise use of generative AI, they share a fundamental lesson: safety claims based on normal inputs alone do not characterize behavior under strategic pressure.

Subsequent literature broadened the focus to the diversity of defenses and to the cybersecurity context. Xu et al. analyzed adversarial attacks on images, graphs and text and demonstrated that the same security risk can manifest in different forms across representations, even though the concrete attack mechanisms differ [8]. Chakraborty et al. reviewed both attacks and defenses, highlighting how quickly offensive and defensive tools evolve and the risk of evaluating a defense against a single attack strategy [9]. Rosenberg et al. studied adversarial machine learning specifically from a cybersecurity perspective and related manipulated learning systems to threat models familiar to security practitioners [10]. This stream of work supports an architectural rather than a model-only response, since deployment security depends on how a model receives information, which services it may call and what authority its outputs carry.

Poisoning research adds the training and supply-chain dimension. The survey by Wang et al. categorized poisoning techniques according to the attacker's objectives, knowledge, capabilities and the defensive strategies involved [6]. Their taxonomy makes it evident that training integrity cannot be assumed when data or learning services are gathered from sources that are only partially trusted. Yerlikaya and Bahtiyar used empirical analysis to show that several learning algorithms across domains can be degraded by poisoning, which confirms that resilience depends on the choice of data and model [7]. In AIOps, similar exposure arises when historical events, operational records, telemetry streams or feedback loops are used to refine automated reasoning. A contaminated incident repository can bias subsequent diagnoses, and malicious

feedback can reinforce unsafe advice. Provenance, validation and controlled update promotion therefore become key components of the security boundary.

Recent work on large language models focuses on instruction control, context boundaries and application integration rather than numerical perturbation. Yao et al. examined the positive and negative security implications of large language models and outlined weaknesses that arise throughout development and use [5]. Chen et al. analyzed how large language models can be used to detect cyber threats and showed that they can improve the interpretation of threat information but must be carefully integrated into security processes [11]. Zhou et al. surveyed security risks related to large language models, including misuse, attacks, privacy and the broader governance challenges posed by large general-purpose models [12]. Kwon and Pak explored methods for performing prompt injection and showed that text can be deliberately crafted to steer model behavior [4]. A key distinction highlighted in this literature, compared with more traditional adversarial settings, is that language-model applications may place trusted policy and untrusted content in the same context channel.

3. METHODOLOGY

The experiment used a controlled design to measure how a layered security architecture affects the behavior of generative-AI and AIOps services under four adversarial conditions. A purpose-built dataset of 172 instances was created, consisting of forty-three instances of each class: prompt injection, data poisoning, adversarial input manipulation and supply-chain tampering. Prompt-injection cases included direct instruction-override attempts, indirect instructions embedded in retrieved text, context-confusion prompts and tool-abuse requests. Poisoning instances included contaminated operational records, misleading feedback records, falsified knowledge snippets and corrupted fine-tuning examples. Adversarial-input cases presented semantically deceptive yet believable content intended to manipulate classification, prioritization or recommendation behavior.

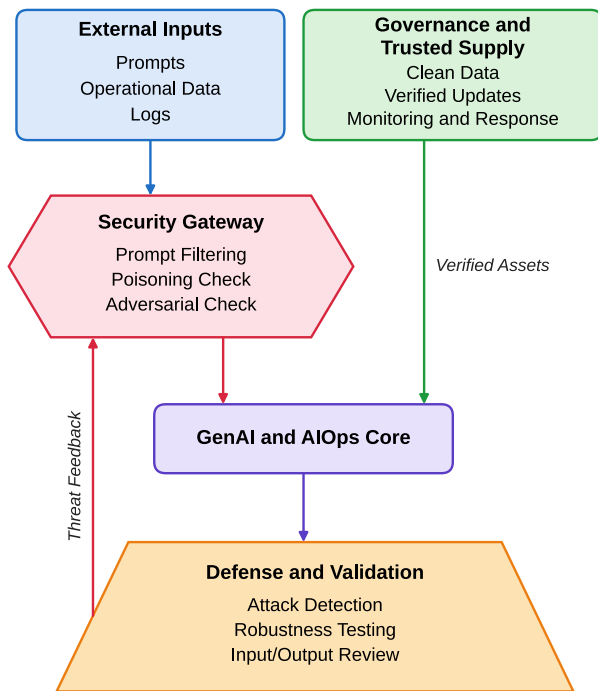


Figure 1. Secure generative-AI and AIOps system architecture.

Prompt injection, data poisoning and adversarial manipulation can compromise generative-AI and AIOps environments, but a compact security architecture can help protect against such attacks, as shown in Figure 1. The first protective layer is the Security Gateway, which receives External Inputs in the form of prompts, operational data and system logs. The gateway performs prompt filtering, poisoning checks and adversarial checks before information is passed to the GenAI and AIOps Core. This core performs intelligent automation, reasoning, monitoring and operational decision support based only on screened inputs. Governance and Trusted Supply provides clean data, verified model updates, monitoring rules and response capabilities, so that the core operates on trusted information. After processing, the core interfaces with the Defense and Validation component, which performs attack detection, robustness testing and input/output review. This layer assesses whether generated responses, operational suggestions or automated actions remain valid when the system is under attack or has been manipulated. Detected threats are fed back to the Security Gateway through the threat-feedback path, so that filtering and validation rules are continually improved. The architecture thus integrates preventive screening, trusted model governance, and runtime validation with feedback-driven adaptation. The design keeps the number of additional components small while supporting safe, resilient and continuously monitored generative-AI and AIOps operations in modern enterprise infrastructure.

Supply-chain cases comprised unauthorized model artifacts, manipulated configuration packages, unverified updates and compromised retrieved content. Every case was executed under three security levels. The baseline condition allowed the AI service to process inputs with normal application validation and no specific adversarial defenses. The gateway condition added prompt inspection, content normalization, input classification, provenance checks and constraints on recognizable tool-abuse patterns. The layered condition added data-integrity validation, signed model and configuration promotion, policy-guided tool

authorization, output validation, adversarial testing, continuous telemetry and incident-response hooks. Each run recorded whether the attack goal succeeded, whether suspicious activity was detected, whether harmful activity was contained before execution, whether benign requests were preserved and whether examinable evidence was retained. The same records and manipulated artifacts were replayed across all configurations to keep comparisons consistent. Test order was randomized within each class to minimize sequence effects, and the environment was reset between runs so that no case could affect subsequent ones. A Python-based harness applied the security logic, issued requests to each of the three configurations and captured structured event traces. Graphs were produced with Matplotlib, and summary statistics were computed from the collected data. No production customer data, personal information or live credentials were used. The study deliberately focused on architectural behavior rather than on comparing models or vendors; the results therefore describe the effect of layered controls and do not claim that any particular foundation model is superior. To avoid the unrealistic assumption that all malicious inputs are easy to recognize, several examples were written to resemble normal operational troubleshooting, documentation or incident text. The tests also included benign operational requests after filtering, so that a defense could not be counted as successful simply by blocking all requests. Detection was counted only when the security layer generated an actionable signal associated with the malicious instance, and containment was counted only when the intended harmful behavior was prevented before it reached a sensitive operation. Recovery readiness was defined as the availability of adequate logs, provenance evidence and policy context to support investigation and controlled restoration. This approach therefore treats security as an end-to-end property spanning input processing, trusted information, model execution, action authorization, monitoring and response.

4. DATA DESCRIPTION

The experimental data comprises 172 controlled security trials developed specifically for this experiment. The data is evenly distributed across four attack types, with 43 instances each of prompt injection, data poisoning, adversarial input manipulation and supply-chain tampering. Each record contains the attack class, an instance identifier, an attack-intensity score, an attack-complexity score, a measure of baseline exposure, the residual risk after gateway protection and the overall residual risk after the full layered defense. The prompt-injection category includes direct overrides, indirect retrieval injection, hidden instructions, context redirection and attempts to make a tool act without authorization. Poisoning records include polluted operational instances, false feedback, distorted knowledge records and biased update material. Adversarial-input records contain plausible but intentionally misleading content designed to manipulate triage, classification or recommended actions. Supply-chain cases include unverified model artifacts, altered configuration packages, manipulated retrieval sources and unauthorized model updates. The logs are synthetic and contain no personal data, customer tracking data or proprietary information. Values were obtained through a reproducible test procedure, so that every instance could be re-run under the baseline, gateway-only and layered-defense conditions. This design allows security behavior to be compared directly while avoiding the privacy and confidentiality concerns associated with live enterprise incident repositories.

5. RESULTS

The evaluation shows that security posture varies considerably when defenses are distributed along the entire AI operating path

rather than concentrated at the user interface. In the baseline environment, malicious instructions often reached the reasoning layer with enough context to alter the response or the recommended action. Direct injection attempts were more readily recognized, whereas indirect attempts embedded in supporting material exposed a more serious weakness, because the application treated retrieved content as valuable context and passed it to the model without a clear trust distinction. This behavior suggests that a generative system cannot safely assume that content obtained through internal retrieval can be treated as trusted instructions. The same applies to AIOps summaries built from tickets, logs and documentation, since any of these may contain adversarial language. The prompt-injection risk can be modeled mathematically as:

$$R_{PI} = \sum_{i=1}^n p_i(\alpha S_i + \beta C_i + \gamma E_i), \alpha + \beta + \gamma = 1 \quad (1)$$

Data-poisoning influence can be expressed as:

$$I_{DP} = \frac{1}{N} \sum_{i=1}^N \|f_{\theta}(x_i) - f_{\theta'}(x_i)\|_2^2 + \lambda \frac{N_p}{N} \quad (2)$$

Table 1. Attack success across security configurations

Attack Class	Instances	Baseline Success (%)	Gateway Success (%)	Layered Success (%)
Prompt Injection	43	79	35	12
Data Poisoning	43	72	40	15
Adversarial Input	43	76	32	10
Supply-Chain Tampering	43	68	38	13

Table 1 summarizes the results of the controlled attacks by class. The full dataset of 172 instances contains 43 instances of each threat class. The highest baseline exposure is for prompt injection at 79 percent, followed by adversarial input at 76 percent, data poisoning at 72 percent and supply-chain tampering at 68 percent. Gateway defenses reduce these rates to 35, 32, 40 and 38 percent respectively, which shows their strong immediate effect on attacks that traverse the input path. The layered architecture achieves a further reduction to 12 percent for prompt injection, 10 percent for adversarial input, 15 percent for poisoning and 13 percent for supply-chain tampering. Poisoning and supply-chain cases show the largest difference between gateway and layered performance, because these threats require data-integrity and deployment controls that input inspection alone cannot provide. The central architectural finding is therefore that wider control coverage yields more uniform protection against attacks from heterogeneous origins. Robustness to adversarial perturbation can be formulated as:

$$\theta^* = \arg \min_{\theta} \frac{1}{N} \sum_{i=1}^N \max_{\| \delta_i \|_p \leq \epsilon} L((x_i + \delta_i)y_i) \quad (3)$$

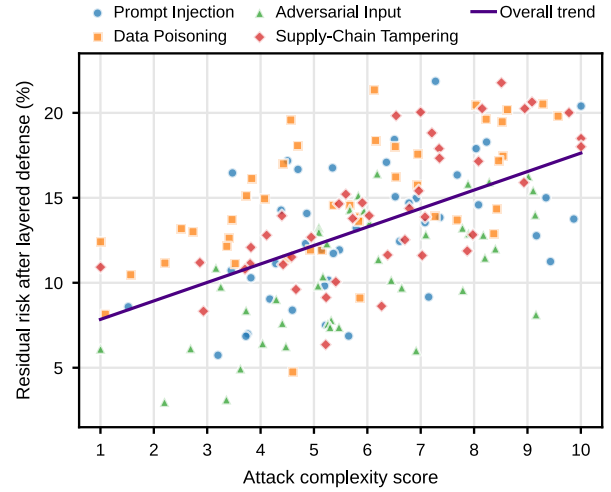


Figure 2. Attack complexity versus residual risk after layered defense.

Figure 2 plots all 172 controlled instances by attack-complexity score and residual risk under the full layered defense. The points cover four attack classes and show that residual risk remains relatively low even as the complexity of the malicious input increases. Across the dataset, residual risk ranges from 3.0 percent to about 21.9 percent, with a mean of about 13.2 percent. The fitted trend rises by about 1.09 percentage points per unit increase in attack complexity, indicating that more sophisticated cases still create additional exposure, but the trend is constrained by the independent controls surrounding the model. Prompt-injection points fall in a band similar to the other classes, which indicates that the architecture does not rely on text filtering alone. Data-poisoning and supply-chain cases account for some of the larger residual values, because these attacks can originate outside the immediate gateway and therefore depend more on provenance checks, update validation and downstream policy checks. Adversarial-input instances are spread across the range, since semantically plausible manipulation tests both the model and the validation logic. Two conclusions follow from the graph: layered protection prevents complexity from translating directly into uncontrolled exposure, and residual risk does not disappear. More complex cases therefore call for ongoing monitoring, expanded red-teaming and periodic policy revision rather than a one-time security deployment. Multi-layer AI security risk can be represented as:

$$R_{AI} = \sum_{j=1}^m w_j P_j V_j I_j + \lambda \sum_{j=1}^m \sum_{k=1, k \neq j}^m \rho_{jk} R_j R_k \quad (4)$$

Table 2 reports operational security quality under the layered framework. Detection recall is 94 percent for prompt injection, 91 percent for data poisoning, 95 percent for adversarial input and 92 percent for supply-chain tampering. False-alert rates remain low at 6, 8, 5 and 7 percent respectively, which shows that the system does not achieve high detection by flagging most activity. Containment effectiveness is 93, 89, 94 and 91 percent for prompt injection, poisoning, adversarial input and supply-chain cases respectively. Recovery readiness is also strong: 90 percent for prompt injection, 88 percent for poisoning, 91 percent for adversarial input and 89 percent for supply-chain tampering. Across these measures, poisoning is the most difficult category, because distorted data can keep a normal structure and continue to have an effect after initial ingestion. The aggregated values indicate that the framework supports both prevention and post-event activities through useful evidence, scoped authorization and traceable policy decisions.

Table 2. Layered security detection, containment and recovery measures

Security Metric	Prompt Injection	Data Poisoning	Adversarial Input	Supply Chain
Detection Recall (%)	94	91	95	92
False Alert Rate (%)	6	8	5	7
Containment Effectiveness (%)	93	89	94	91
Recovery Readiness (%)	90	88	91	89

Malicious-input detection can be modeled mathematically as:

$$D(x) = \sigma \left(b + \sum_{j=1}^m w_j z_j(x) \right), \sigma(u) = \frac{1}{1+e^{-u}} \quad (5)$$

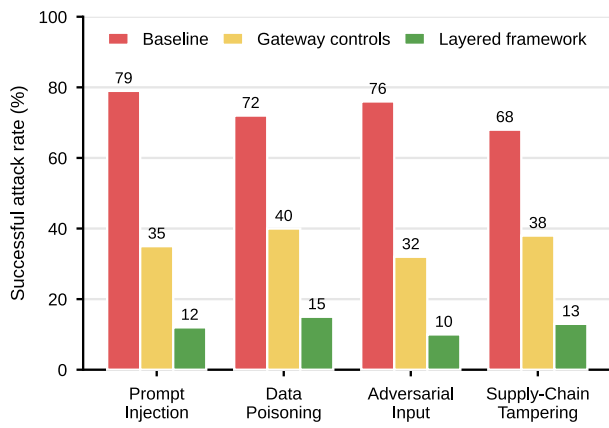


Figure 3. Attack success rates of the four attack classes under each security configuration.

Figure 3 compares the attack success rates of the four attack classes under the baseline, gateway-only and full layered configurations. Prompt injection falls from 79 percent at baseline to 35 percent with gateway controls and to 12 percent with the layered design. Data poisoning starts at 72 percent, drops to 40 percent with gateway protection and to 15 percent once provenance, validation and governance controls are introduced. Adversarial input falls from 76 percent to 32 percent and then to 10 percent, the lowest final rate of the four classes. Supply-chain tampering begins at 68 percent, decreases to 38 percent with gateway controls and reaches 13 percent with signed update paths and policy checks. The trend shows why filtering at the point of input is not sufficient. Gateway defenses reduce the effectiveness of direct, input-oriented attacks, but they offer less protection against poisoning and supply-chain threats, which can enter through a legitimate data or deployment path. The layered structure closes these gaps by placing controls on data, model artifacts, outputs and actions. The final success rates indicate that no single class is eliminated entirely; however, keeping all classes at a low residual level yields a more manageable operational risk profile and leaves fewer events for monitoring and incident response to investigate. The overall security effectiveness of generative-AI and AIOps systems can be quantified as:

$$S_{eff} = \omega_1 D_r + \omega_2 P_r + \omega_3 A_r + \omega_4 G_r - \omega_5 F_r, \sum_{j=1}^5 \omega_j = 1 \quad (6)$$

Gateway controls improved the handling of overtly malicious patterns and reduced successful manipulation through direct instruction override. Most attacks did not reach the core service

in their original form because of normalization, request inspection, content classification and early provenance checks. Nevertheless, the gateway alone remained vulnerable when adversarial behavior relied on information that appeared legitimate in isolation. A corrupted knowledge record may pass input screening because of its normal structure, and a compromised model package may never pass through the prompt gateway at all. The intermediate configuration therefore confirmed the need for front-door security, but it cannot withstand attacks that originate deeper in the data or model lifecycle.

The complete layered structure produced the most consistent defensive behavior, because every attack had to pass through several independent decision points. Suspicious instructions were screened or isolated before model execution. Provenance controls detected untrusted or modified supporting material. Output verification checked whether generated recommendations conflicted with expected operational policy. Tool authorization prevented the language model from turning a manipulated response directly into a privileged action. Monitoring tracked recurring suspicious patterns and linked them to the originating input, data source and policy decision. Together, these controls reduced the likelihood that a single missed indicator would lead to a full compromise.

Prompt-injection cases illustrated the benefit of separating content from authority particularly clearly. When the model received retrieved text containing hostile instructions, the layered design caused the document to be treated as evidence rather than as a command. The model could still summarize the useful part of the document, while the security logic blocked attempts to redefine system goals or to request restricted actions. This preserves utility, because legitimate retrieval remains available, but the trust level of retrieved content does not change silently. Attempts to hide injected instructions within ordinary troubleshooting language were also easier to handle when the monitors considered instruction hierarchy and the capability of the requested tool together with suspicious keywords, rather than matching keywords alone.

6. DISCUSSION

The results indicate that securing generative-AI and AIOps systems requires a shift from model protection to trust-boundary engineering. Table 1 reports the baseline attack success rates and shows that every threat class evaluated can have a meaningful impact on an insufficiently protected system. The success rate is 79 percent for prompt injection, 76 percent for adversarial input, 72 percent for data poisoning and 68 percent for supply-chain tampering. These values should not be read as measures of attack severity, but as evidence of how much trust the system places in the model and how comparable outcomes can arise from different entry points. An attacker does not need to compromise the same layer in every scenario: one adversary may modify the prompt, another may alter data and a third may insert a malicious artifact. A secure architecture should therefore ensure that none of these paths leads uncontrolled to operational authority.

Gateway protection alone produces a visible improvement. Prompt injection drops from 79 percent to 35 percent and adversarial input from 76 percent to 32 percent, which confirms the value of early inspection. However, data poisoning remains at 40 percent and supply-chain tampering at 38 percent, because a prompt gateway does not necessarily verify historical data or deployed components. This imbalance matters because enterprise teams may be tempted to deploy a single LLM firewall and treat it as a complete solution. Such a gateway can reduce overtly malicious instructions, but its protection does not

automatically extend to retrieval indexes, fine-tuning sets, model registries, update pipelines or feedback loops. Under the full layered architecture, the success rates for prompt injection, data poisoning, adversarial input and supply-chain tampering fall to 12, 15, 10 and 13 percent respectively. This is the largest reduction, because each attack must overcome multiple checks. A prompt-injection request can be blocked at the point of entry, its retrieval source can be marked as untrusted, its output can be checked against policy and the requested tool call can still require approval. An attacker must therefore do more than persuade the model; the malicious payload must also evade source controls and action limits. As a result, a successful jailbreak or injection has less impact, because downstream controls limit its effect.

Figure 3 illustrates this pattern across the three security states. The baseline bars are uniformly high, the gateway bars are lower but still significant, and the layered bars are clustered near the bottom. The pattern shared by all four classes matters more than any single category, because it shows that the architecture improves consistency. This uniformity helps security teams prioritize residual events, since no single threat family is left markedly less protected than another. A control plan that excelled only against prompt injection could unintentionally push attackers toward poisoning or supply-chain paths; a balanced reduction guards against such displacement.

Figure 2 adds a second dimension: attack complexity. Residual risk across the 172 instances ranges from 3.0 to approximately 21.9 percent, with a mean of nearly 13.2 percent. The overall trend increases by approximately 1.09 percentage points per unit increase in complexity. This positive slope is expected, because more advanced attacks involve ambiguous, concealed and cross-layer behavior. The slope is nonetheless modest: additional complexity yields only a limited gain for the attacker, because even the most complex cases remain bounded by controls that do not depend on the model recognizing every attack. This reflects a pragmatic approach to security, in which controls are designed to limit the impact of model uncertainty instead of assuming that the model can reliably detect all malicious instructions.

The operational metrics in Table 2 also show why detection quality must be considered together with false alerts and containment. Detection recall ranges from 91 to 95 percent and false-alert rates from 5 to 8 percent across attack categories. A system that achieved similar recall by flagging almost every prompt would be costly to review and would encourage users to circumvent the controls. The low false-alert levels observed here indicate that the security logic can remain selective without disrupting production use. When manipulated content enters the reasoning stage, adversarial input shows the highest detection recall (95 percent) and the lowest false-alert rate (5 percent), which suggests that behavioral validation and contextual checks are effective.

Containment is equally critical, because a detected attack can still cause harm if the model is allowed to operate without restriction. Containment effectiveness is 93 percent for prompt injection, 89 percent for poisoning, 94 percent for adversarial input and 91 percent for supply-chain tampering. These values indicate the safety margin that authorization and policy gates provide after detection. In AIOps, this margin can determine whether a suspicious recommendation waits in a review queue or an automated remediation changes a live service. The architecture deliberately separates execution privilege from reasoning capability, so that the model remains useful even when it produces a wrong or manipulated answer, while the blast radius of the error or manipulation is limited.

Recovery readiness ranges from 88 to 91 percent. These values matter because some attacks will evade preventive measures, and organizations need sufficient information to determine what has changed. Poisoning sits at the lower end of the range at 88 percent, owing to the difficulty of establishing when a poisoned record entered a data source and which subsequent decisions relied on it. Adversarial input scores highest at 91 percent, because the event can be tied to a specific request and correlated more closely with validation and response logs. This gap indicates that data lineage requires further investment. Improved versioning, approval histories, immutable sources and rollback capability would strengthen both the prevention of and the recovery from poisoning.

The results also support a zero-trust view of AI context. Retrieved documents, operational tickets and system log files should not automatically inherit the trust of the application that displays them, and the same applies to model-generated summaries. Every source should carry a statement of provenance and a statement of how it may be used. A document may supply information but should not act as an executable instruction. A model update may be technically correct yet still require an approved promotion path. A coherent generated remediation does not by itself confer the authority to execute. Separating these decisions prevents the architecture from concentrating security decisions in a single probabilistic component. The reported figures can also help governance teams make measurable statements instead of sweeping claims about protecting an AI system. Attack success, detection recall, false-alert rate, containment effectiveness and recovery readiness express different security objectives and should be measured separately. The number of successful attacks may fall while the number of false alarms rises, which can reduce operational usefulness. High detection combined with weak containment can still allow damaging actions. Effective prevention combined with ineffective recovery can leave an organization unable to gauge the extent of a successful compromise. The layered approach is effective because it addresses all of these aspects within a single design.

7. CONCLUSION

This study tested 172 controlled instances of prompt injection, data poisoning, adversarial input manipulation and supply-chain tampering to evaluate a multi-layered security approach for generative-AI and AIOps systems. The results indicate that an input gateway offers valuable protection but is not sufficient against attacks that arrive through trusted data repositories, model updates or deployment pipelines. The full architecture reduces successful attacks to 12 percent for prompt injection, 15 percent for poisoning, 10 percent for adversarial input and 13 percent for supply-chain tampering, all significantly lower than the baseline exposure. It also delivers strong operational performance, with detection recall of 91–95 percent, containment of 89–94 percent, recovery readiness of 88–91 percent and false-alert rates of around 5–8 percent. The most significant conclusion is architectural: the more clearly a system separates untrusted content, instruction authority, model reasoning and execution, the stronger it becomes. Each of the following provides a distinct layer of protection: provenance checking, controlled promotion of updates, adversarial validation, scoped tool permissions, continuous monitoring and evidence for incident response. The scatter analysis also shows that more advanced attacks leave higher residual exposure, so AI security should be treated as an evolving control system rather than a one-time solution. The recommended posture for enterprise deployments is therefore defense in depth across the entire AI lifecycle. Generative models can deliver considerable

operational benefit, but their context, data sources, artifacts and actions should be independently verified before they inform high-impact AIOps decisions.

8. REFERENCES

- [1] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognition*, vol. 84, pp. 317–331, 2018. <https://doi.org/10.1016/j.patcog.2018.07.023>
- [2] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, vol. 6, pp. 14410–14430, 2018. <https://doi.org/10.1109/ACCESS.2018.2807385>
- [3] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 9, pp. 2805–2824, 2019. <https://doi.org/10.1109/TNNLS.2018.2886017>
- [4] H. Kwon and W. Pak, "Text-based prompt injection attack using mathematical functions in modern large language models," *Electronics*, vol. 13, no. 24, art. 5008, 2024. <https://doi.org/10.3390/electronics13245008>
- [5] Y. Yao, J. Duan, K. Xu, Y. Cai, Z. Sun, and Y. Zhang, "A survey on large language model (LLM) security and privacy: The Good, The Bad, and The Ugly," *High-Confidence Computing*, vol. 4, no. 2, art. 100211, 2024. <https://doi.org/10.1016/j.hcc.2024.100211>
- [6] Z. Wang, J. Ma, X. Wang, J. Hu, Z. Qin, and K. Ren, "Threats to training: A survey of poisoning attacks and defenses on machine learning systems," *ACM Computing Surveys*, vol. 55, no. 7, art. 134, 2023. <https://doi.org/10.1145/3538707>
- [7] F. A. Yerlikaya and Ş. Bahtiyar, "Data poisoning attacks against machine learning algorithms," *Expert Systems with Applications*, vol. 208, art. 118101, 2022. <https://doi.org/10.1016/j.eswa.2022.118101>
- [8] H. Xu, Y. Ma, H.-C. Liu, D. Deb, H. Liu, J.-L. Tang, and A. K. Jain, "Adversarial attacks and defenses in images, graphs and text: A review," *International Journal of Automation and Computing*, vol. 17, pp. 151–178, 2020. <https://doi.org/10.1007/s11633-019-1211-x>
- [9] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "A survey on adversarial attacks and defences," *CAAI Transactions on Intelligence Technology*, vol. 6, no. 1, pp. 25–45, 2021. <https://doi.org/10.1049/cit2.12028>
- [10] I. Rosenberg, A. Shabtai, Y. Elovici, and L. Rokach, "Adversarial machine learning attacks and defense methods in the cyber security domain," *ACM Computing Surveys*, vol. 54, no. 5, 2021. <https://doi.org/10.1145/3453158>
- [11] Y. Chen, M. Cui, D. Wang, Y. Cao, P. Yang, B. Jiang, Z. Lu, and B. Liu, "A survey of large language models for cyber threat detection," *Computers & Security*, vol. 145, art. 104016, 2024. <https://doi.org/10.1016/j.cose.2024.104016>
- [12] W. Zhou, X. Zhu, Q.-L. Han, L. Li, X. Chen, S. Wen, and Y. Xiang, "The security of using large language models: A survey with emphasis on ChatGPT," *IEEE/CAA Journal of Automatica Sinica*, vol. 12, no. 1, pp. 1–26, 2025. <https://doi.org/10.1109/JAS.2024.124983>