LLMZ+: Contextual Prompt Whitelist Principles for Agentic LLMs

Tom Pawelek[1], Raj Patel[2], Charlotte Crowell[2], Noorbakhsh Amiri Golilarz[2],
Sudip Mittal[2], Shahram Rahimi[2], and Andy Perkins[1]

[1]Mississippi State University, Mississippi State, MS, USA
[2]The University of Alabama, Tuscaloosa, AL, USA

***Abstract*—Compared to traditional models, agentic AI represents a highly valuable target for potential attackers as they possess privileged access to data sources and API tools, which are traditionally not incorporated into classical agents. Unlike a typical software application residing in a Demilitarized Zone (DMZ), agentic LLMs consciously rely on nondeterministic behavior of the AI (only defining a final goal, leaving the path selection to LLM). This characteristic introduces substantial security risk to both operational security and information security. Most common existing defense mechanism rely on detection of malicious intent and preventing it from reaching the LLM agent, thus protecting against jailbreak attacks such as prompt injection. In this paper, we present an alternative approach, LLMZ+, which moves beyond traditional detection-based approaches by implementing prompt whitelisting. Through this method, only contextually appropriate and safe messages are permitted to interact with the agentic LLM. By leveraging the specificity of context, LLMZ+ guarantees that all exchanges between external users and the LLM conform to predefined use cases and operational boundaries. Our approach streamlines the security framework, enhances its long-term resilience, and reduces the resources required for sustaining LLM information security. Our empirical evaluation demonstrates that LLMZ+ provides strong resilience against the most common jailbreak prompts. At the same time, legitimate business communications are not disrupted, and authorized traffic flows seamlessly between users and the agentic LLM. We measure the effectiveness of approach using false positive and false negative rates, both of which can be reduced to 0 in our experimental setting.**

***Index Terms*—security, LLM, Prompt injection, Prompt jailbreak, Agentic AI, LLMZ+, Prompt whitelisting, Contextual whitelisting**

I. Introduction

Throughout commercial and academic applications, there are numerous products, studies and libraries designed to prevent malicious prompts from interacting with LLMs in production environments. All of these approaches operate on the principle of identifying malicious messages and blocking their progression within the system. The inherent structure is similar to that of traditional anti-malware products, relying on a predefined set of signatures and heuristics. Since all detection is driven by these definition databases, they must be updated regularly. When a new attack technique emerges, the definitions must be revised to ensure that the system can recognize the new threat.

These products, therefore, introduce additional capital expenditures (CapEx) [1] and operating expenses (OpEx) [1], as their maintenance requires specialized resources. They also pose a hidden risk of "failing silently", whereby outdated or delayed definition updates can result in the system providing a false sense of security while remaining vulnerable to newly developed attacks. Our objective was to design a framework that is more accommodating for administrators and is based on a security principle that leverages the contextual information of the subject being served by the agentic LLM.

In developing our solution, we drew inspiration from the most secure practices employed in perimeter firewall design. Rather than maintaining an exhaustive database of all global threats (sources, payloads, etc.), the recommended strategy is to allow only those types of traffic that are explicitly recognized as safe and legitimate, while blocking all other network packets [2]. As a result, a typical configuration specifies the scenarios in which access is permitted (such as traffic originating from domestic IP addresses, residential internet service providers, or specific destination ports) which is then concluded with a catch-all rule: $DENY\,ALL$.

This approach relieves network administrators from the burden of maintaining an ever-growing list of foreign addresses, Virtual Private Network (VPN) [3] subnets, exploited ports and daemon processes [4], and and other threat definitions. Instead, the system operates on the principle that only recognized and compliant traffic is permitted. We adopt this foundational principle and apply it to the design of our LLM guard, as described in the following sections.

In this paper, we present *LLMZ+*: a solution designed to safeguard agentic AI models against prompt injection attacks. Prompt injection is a form of jailbreak attack in which malicious prompts are used to bypass built-in checks and override the control mechanisms of LLMs [5]. Our approach introduces a contextual whitelisting mechanism that is grounded in a comprehensive understanding of analyzed messages, ensuring that only those prompts relevant to the intended use case are permitted. The major contributions of this paper are as follows:

- We introduce ***LLMZ+*** (LLM + DMZ [6]) as a conceptual security boundary for agentic LLMs, drawing inspiration

from the Demilitarized Zone (DMZ) architecture in network security.
- We highlight the limitations of conventional prompt threat detection and mitigation techniques, which often rely on static heuristics and often struggle to counter adaptive adversaries.
- We introduce a framework tailored for business-focused agentic LLMs, designed for both operational reliability and security resilience.
- We evaluate LLMZ+ using a benchmark of documented prompt injection attacks, alongside authentic business communications submitted to agentic chatbots.

The remainder of this paper is structured as follows. Section II reviews the current state-of-the-art and related work. Sections III and IV introduce the LLMZ+ framework and detail the threat model addressed in this study. Sections V through VII present our experimental setup, results, and practical considerations for deploying LLMZ+ in production business environments. Finally, Section VIII offers conclusions and outlines directions for future research.

II. Background and Related work

In this section, we provide the background on previously explored jailbreaking techniques within LLM as well as defenses that have been presented to combat against it. The phenomenon of jailbreaking in LLMs refers to the circumvention of built-in safety mechanisms by crafting adversarial prompts that elicit responses normally restricted by the system's alignment objectives [7]. Early work by Wei et al. [8] introduced Greedy Coordinate Gradient Ascent (GCG), a method using suffix-based mismatch objectives and alternate encodings (e.g., base64) to bypass safety filters. Building on this, Zou et al. [9] extended the attack by combining multi-prompt, multi-model strategies to create universal jailbreak prompts.

Several studies have analyzed the taxonomy of jailbreak strategies. Liu et al. [7] categorized jailbreak prompts into three core patterns: pretending, attention shifting, and privilege escalation. Their study found that both ChatGPT-3.5 and GPT-4.0 were vulnerable to these methods, with an 86% success rate. Gupta et al. [10] identified four dominant jailbreak vectors: role-playing (e.g., Do Anything Now (DAN) or Developer Mode), reverse psychology, model escape, and prompt injection. These techniques can be exploited to generate content for malicious purposes such as phishing, social engineering, and malware development. Similarly, Yu et al. [11] conducted a qualitative study involving user-generated prompts, showing that even untrained users were capable of producing effective jailbreaks. Their work also introduced a hybrid human-AI prompting framework, though they noted the AI component's difficulty in adapting to semantic nuance.

Beyond prompt engineering, Carlini et al. [12] demonstrated that brute-force strategies and adversarial multi-modal inputs (e.g., malicious images) could induce harmful outputs, suggesting a broader surface for attack. Complementing these studies, Chen et al. [13] proposed a suffix-classification model that successfully identified and blocked GCG-style jailbreaks with 96% accuracy.

The challenges extend to multilingual and synthetic language models as well. Deng et al. [14] found that LLMs are more vulnerable in low-resource languages, while Oh et al. [15] showed that malicious prompts encoded in synthetic ASCII-based formats could also bypass content filters.

From a system-wide perspective, Yao et al. [16] conducted a large-scale survey on the intersection of LLMs and security, categorizing existing literature into three thematic areas: the use of LLMs for defensive cybersecurity applications ("good"), the misuse of LLMs for offensive purposes such as cyberattacks ("bad"), and studies addressing inherent vulnerabilities in LLMs along with corresponding defense mechanisms ("ugly"). Separately, Das et al. [17] provided an in-depth analysis of LLM-specific vulnerabilities, including jailbreaking, data poisoning, and personally identifiable information (PII) [18] leakage, and surveyed a broad range of proposed mitigation strategies.

Among emerging defenses, Reinforcement Learning from Human Feedback (RLHF) has been foundational in improving LLM alignment [19], with Ganguli et al. [20] further emphasizing red teaming as a critical evaluation method. Additional efforts like *SmoothLLM* [21], *LLM Guard* by Protect AI [22], and Deng et al.'s *SELF-DEFENCE* framework [14] employ strategies such as prompt sanitization, multilingual safety data generation, and adversarial fine-tuning. Hila et al. [23] proposed reducing prompt perplexity through translation by a secondary LLM to improve resilience.

Despite these efforts, a consistent pattern emerges in that most defenses rely on either input and output prompt filtering or RLHF-based alignment techniques. Input filtration typically targets jailbreak attempts, while output filtration aims to prevent the disclosure of Personally Identifiable Information (PII), or other sensitive content. These approaches often depend on static mechanisms, such as maintaining and updating keyword lists or periodically retraining models, which may limit their adaptability and responsiveness. RLHF, while effective, is costly, time-consuming, and requires frequent retraining to remain relevant against evolving attack strategies.

By contrast, our approach, termed **LLMZ+**, is highly restrictive in its behavior, continuously context-aware, and capable of dynamically enforcing alignment during inference. It actively monitors whether a prompt deviates from the operational domain, and any deviation triggers an immediate denial of the response. This persistent, context-aware, real-time validation distinguishes our framework from prior defenses that depend on static filters or periodic model updates. In the next section, we elaborate on the core architecture and enforcement logic underpinning the LLMZ+ framework.

III. Principles of LLMZ+

In our context-based approach, we employ a Guard Prompt, as depicted in Fig. 1, to evaluate all incoming external messages. Instead of searching for prompt exploits, which may include a variety of linguistic and mathematical manipulations,

our method ensures that every message is fully understood by the guarding LLM and corresponds to the expected use case of the ongoing conversation. To further protect outbound messages produced by the agentic AI, an information scope layer can be incorporated. This additional protection may be integrated directly within the LLM or implemented through established Data Loss Prevention (DLP) mechanisms [24]. The information scope explicitly defines which categories of information, such as specific types of PII, the model is authorized to disclose. For instance, this restriction may only permit the model to return information that is necessary for a customer to access their account.

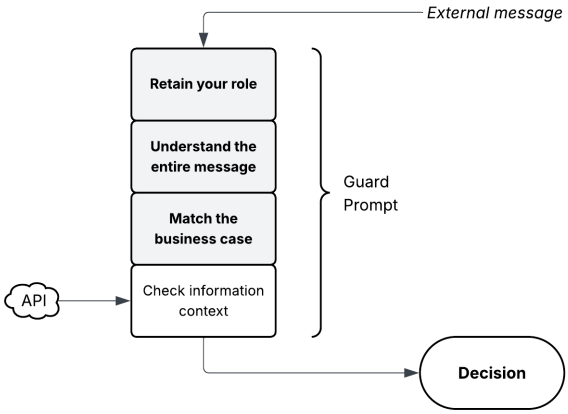

Fig. 1: LLMZ+ Guard Prompt Structure

This approach protects the agentic LLM from a wide range of exploitive prompts and mitigates the risk of both known and novel prompt injection attacks. By leveraging the specific deployment context of the agent, the scope of acceptable content exchanged with the agentic core is further narrowed, reducing the potential attack surface.

The result of the Guard Prompt evaluation may be represented as either a binary decision indicating yes or no, or as a quantified risk score ranging from 0 to 10 for more nuanced applications. Ideally, all malicious messages would be assigned a risk score of 10, while all benign messages would receive a score of 0.

IV. Threat Model

In this section, we define the specific threat model targeted by the LLMZ+ framework. We begin by describing typical deployment scenarios for agentic LLMs in practical business contexts. Next, we analyze major attack vectors relevant to these deployments and detail the potential risks they introduce. We then present our proposed security solution, including both its architectural design and deployment strategy. This overview establishes a foundation for understanding the security objectives, effectiveness, and boundaries of the LLMZ+ approach.

A. Typical Agentic LLM Deployment

In our analysis, agentic AI is frequently deployed as a customer-facing LLM (via web chat, phone audio or mobile text). In commercial settings, these agentic models are designed to serve specific use cases, such as:

- Providing customer support,
- Facilitating payments,
- Assisting with product or service selection.

It is important to note that our solution does not attempt to secure generic, all-purpose agents that are publicly accessible and context-agnostic, where the nature of each interaction is determined solely by the end-user. Such systems are usually not hosted within a corporate Demilitarized Zone (DMZ) [6], which is an isolated network segment positioned between internal and external environments and commonly used to provide controlled access to on-premise services. As a result, these agents generally lack privileged access to sensitive data sources or APIs. In contrast, agentic LLMs designed for targeted business tasks often require access to confidential, non-public information in order to complete their assigned task.

B. Attack Vectors

Given the privileged access held by these agentic LLMs, attackers may attempt to exploit prompt engineering techniques to "jailbreak" the model and gain unauthorized control over sensitive information. For an agentic LLM integrated with multiple APIs, manipulating the model to perform arbitrary API calls and reveal the results is analogous to obtaining shell access to a compromised server. The consequences of such breaches are similar to those of traditional network intrusions and typically manifest in two primary forms:

- **Sensitive data leakage:** The compromised LLM may be used to extract personally identifiable information (PII), financial records, trade secrets, or other confidential data [25].
- **Unauthorized activity:** Attackers can induce the LLM to execute actions resulting in material harm, such as transferring funds, sending unauthorized communications, or disrupting system operations.

Although one might assume that the impact is confined to the data and tools directly accessible to the LLM, in practice, these attacks often serve as an initial foothold for further lateral movement within the organization's network. This type of exploitation can bypass traditional security controls, including firewalls and DLP systems.

Attackers typically target the LLM through public interfaces, without requiring any privileged access or insider knowledge of the AI deployment. For the purposes of this study, we restrict our focus to prompt-based attacks and do not address other forms of network or software compromise. Accordingly, LLMZ+ is not intended to replace a comprehensive information security (infosec) architecture [26]; instead, it serves as an additional safeguard that protects the LLM from prompt-based adversarial attacks.

C. Proposed Solution

As shown in Fig. 2, our approach leverages an auxiliary LLM (in blue), to function as a whitelist guard for both ingress prompts sent from external users, as well as egress replies returned by the agentic LLM (in yellow). Following the "Firewall principle", instead of attempting to enumerate and block every possible malicious input, LLMZ+ evaluates each message against a set of strict criteria.

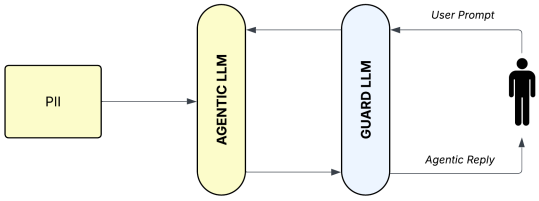

Fig. 2: LLMZ+ Data Flow

The **Ingress filter** verifies that messages received from external users meet the following requirement, such as:

- The message is fully interpretable by the Guard Prompt.
- The message is consistent with a natural customer-service conversation.
- The message is relevant to the business case served by the Agentic LLM.

The **Egress filter** ensures that outbound messages also remain consistent with the intended business use case. To enhance our safety mechanism, we can incorporate a very simple contextual Retrieval-Augmented Generation (RAG), which informs the guard LLM about the categories of data that the user is permitted to access (e.g. account information, balance, transactions, etc.). A simpler variant uses the information whitelisting implemented directly in the prompt or regex-based filter to block the disclosure of sensitive information like Social Security Numbers (SSNs).

Messages in either direction that do not satisfy these criteria are blocked, preventing the exploitation of the agentic LLM through prompt-based attacks. It is important to emphasize that this solution is designed specifically to address prompting threats and does not mitigate risks associated with other layers of the AI deployment stack, such as network security vulnerabilities or traditional software exploits.

D. Deployment Strategy

Our implementation is evaluated in the context of a commercial fintech chatbot [27] deployed within a highly regulated retail market in the US. In this deployment, when a suspicious message is intercepted by LLMZ+, the system notifies a human operator who can then review the communication or directly intervene in the conversation as necessary.

To assess the effectiveness of our approach compared to traditional LLM threat detection methods, we constructed a controlled testing environment based on an on-premises setup that includes the following components:

- 2x Llama3.1 [28] / Llama3.3 [29] models (agentic prompts + guard prompts)
- OpenWebUI [30]
- A static data source and dynamic API accessible to the Agentic LLM

The primary task in this scenario involves performing customer account login and balance confirmation. This represents a very simple use case, albeit it can be implemented in an agentic fashion, using a set of OpenAPI calls depending on how our customers decide to authenticate (account number, SSN or a phone lookup). This scenario is an example where case specificity is used as a foundation of context white-listing. It can be easily generalized to any similar business use.

We conducted two types of evaluations. First, we sent a representative set of legitimate customer messages through LLMZ+ to measure the rate of **false positives** rates. Second, we utilized a public repository of "GPT Super Prompting" [31] techniques, which contains the most recent jailbreak techniques designed to fool LLMs into acting against the constraints defined by their authors. Our goal was to measure how many of those prompts would be blocked by LLMZ+, and how many would pass through (i.e. the **false negative** rates).

Attack attempts against whitelisting-based filters are comparatively rare. key advantage of our approach is the clear separation of user-supplied messages from the rest of the LLM prompt, which makes it significantly more difficult for adversaries to conceal and deliver malicious instructions that might be executed by our agent.

V. Methodology

Our system setup is depicted in Fig. 3 and described in detail in "Deployment Strategy" in the previous sub-section. The primary objective of our study is to minimize false rates, ideally reducing them to zero, which would indicate an optimally configured system with full alignment between quantified and binary evaluation scores. The false positive and false negative ratios were experimentally measured as follows:

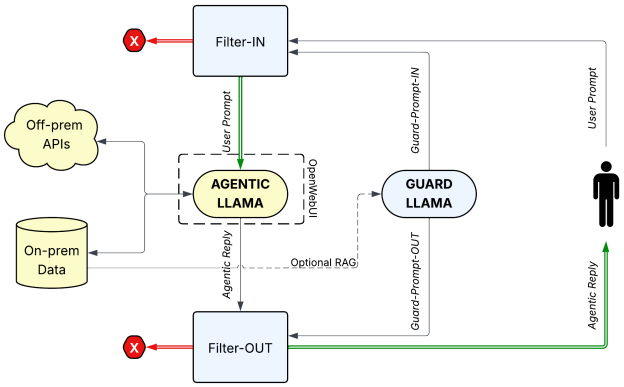

Fig. 3: LLM Infrastructure Setup

A. False Negatives

False negatives are defined as malicious messages that are erroneously allowed to be processed by the agentic LLM. This rate serves as a key indicator of the system's effectiveness in threat mitigation. To specifically evaluate the core performance of LLMZ+, we did not incorporate a Retrieval-Augmented Generation (RAG) [32] in our tests.

A set of scripts was developed to simulate user inputs, utilizing prompts from [31]. These prompts were submitted to a RESTful API endpoint [33], which served as the backend for a text-based chatbot. The false negative rate was calculated as follows:

$$F_{\text{negative}} = 1 - \frac{C_{\text{flagged}}}{C_{\text{total}}} \tag{1}$$

where:

- F_{negative} refers to the false negative rate
- C_{flagged} refers to the count of messages flagged by LLMZ+
- C_{total} refers to the total count of messages posted

B. False Positives

False positives are legitimate messages that do not aim to exploit the agentic LLM, but are incorrectly flagged as malicious by LLMZ+. While not a direct information security threat, a high false positive rate could discourage commercial adoption due to the risk of business disruption and degraded customer service.

To evaluate the false positive rate, we compiled a collection of authentic end-user messages from agentic bots deployed in business environments similar to our experimental context. These scenarios required customers to authenticate in order to access restricted information, such as PII. In these cases, we expect LLMZ+ to allow all legitimate messages to pass through. The false positive rate was determined as follows:

$$F_{\text{positive}} = \frac{C_{\text{flagged}}}{C_{\text{total}}} \tag{2}$$

where:

- F_{positive} refers to the false positive rate,
- C_{flagged} refers to the count of messages flagged by LLMZ+,
- C_{total} refers to the total count of messages posted.

C. Decision Threshold (DT)

For the purpose of our experiment, we define Decision Threshold (DT) as a cut-off value. Any messages assigned a score equal to or greater than DT are stopped by LLMZ+ and are not forwarded to the agentic AI worker.

VI. Results and Observations

To facilitate risk-based decision-making, our experiments employed a quantified output from the guarding LLM. In this framework, the LLM produces a risk score between 0 and 10, allowing system administrators to specify the threshold at which messages are blocked. We conducted experiments using on-premise Llama models with $Llama3.1_{8B}$, $Llama3.3_{70B}$, and $Llama3.1_{405B}$ configurations. Each incoming message was evaluated ten times using the $Llama3.1_{8B}$ model, and three times with the other two models. This repeated evaluation was performed to capture worst-case scenarios, although a single evaluation would generally suffice in production environments. The maximum risk score observed for each message was recorded as the final outcome. For the $Llama3.1_{8B}$ model, as depicted in Fig. 4, the first false negative was observed at a decision threshold of 6. Threshold values of 9 and 10 were found to be insecure, as they allowed an excessive number of malicious messages to pass through.

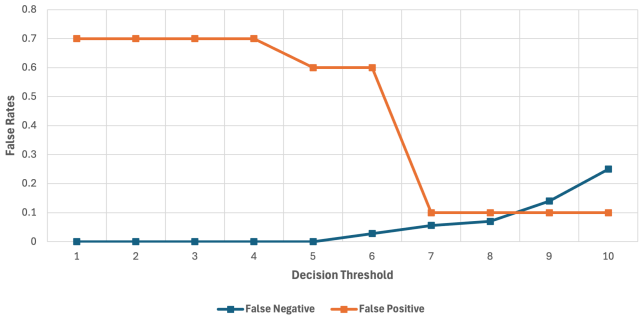

Fig. 4: False rates, $Llama3.1_{8B}$, C_{tot} = 71

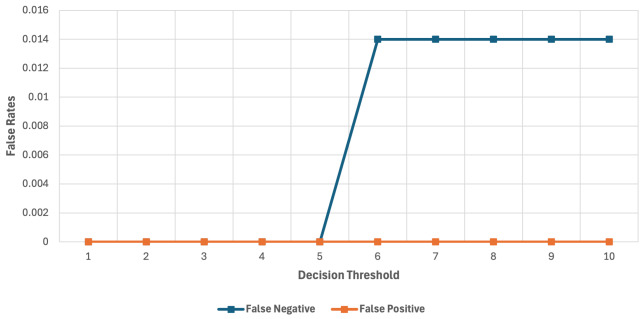

Fig. 5: False rates, $Llama3.3_{70B}$, $Llama3.1_{405B}$, C_{tot} = 71

As anticipated, transitioning from the $Llama3.1_{8B}$ model to the $Llama3.3_{70B}$ model resulted in a marked improvement in detection performance, effectively reducing the false positive rate to zero, as shown in Fig. 5. This enhancement can be attributed to the design of our guard prompt, which requires sufficient model capacity to enforce rigorous message filtering. The evaluation process involves two distinct layers: the guard prompt and the subject message. The overall prompt length, along with the complexity of these layers, may exceed the processing capabilities of the $Llama3.1_{8B}$ model. Nonetheless, there is an optimal range of decision thresholds between 6 and 7 for $Llama3.1_{8B}$, which balances the detection of malicious content with the minimization of false positives. For the $Llama3.3_{70B}$ and $Llama3.1_{405B}$ models, a broader range of thresholds from 1 to 5 yields a false positive rate of zero. This flexibility enables fine-tuning of the pass-through criteria based primarily on the desired level of sensitivity for malicious message detection.

In practical business deployments, LLMZ+ is utilized in combination with a simple message pre-processing step. As described in the following section, this configuration achieves ideal performance, resulting in both false positive and false negative rates of zero across all tested threshold values. Furthermore, we have successfully implemented LLMZ+ with various cloud-based models, including Google's Gemini and its agentic framework.

VII. Practical Considerations

When deployed in a production environment, LLMZ+ must satisfy specific performance requirements to avoid disrupting the end-user experience. As demonstrated in the Results section, both the $Llama3.3_{70B}$ and $Llama3.1_{405B}$ models provide near-perfect detection performance. However, in resource-constrained settings, their execution times may be prohibitive. The following considerations outline methods to improve the efficiency of $Llama3.1_{8B}$ deployments.

A. False Positive Overrides

The majority of false positive cases observed with $Llama3.1_{8B}$ models stem from an incomplete understanding of the guard prompt. In particular, instructions related to associating risk ratings exclusively with prompt attack conditions, rather than with general data sensitivity, tend to be overlooked by models with fewer parameters. When evaluating LLMZ+ against real-world chatbot transcripts, we found that most false positives could be attributed to a limited set of frequently encountered message types, such as social security numbers, dates (including date of birth), and addresses. These messages can be detected and bypassed by a simple non-LLM filter, significantly reducing the false positive rate.

B. Message Pre-processing

Certain characteristics of incoming messages can be efficiently screened using traditional filters. One of the most effective checks is to impose a maximum message length. Prompt injection and related LLM exploit techniques typically require messages of considerable length to encode their malicious instructions. Such instructions often involve complex role redefinitions for the agentic model or the inclusion of encoding and decoding steps that enable the transfer of unauthorized content. By limiting the permissible message length, the system can block the vast majority of these attacks. In our experiments, a combination of message length filtering and use of the $Llama3.3_{70B}$ model resulted in both false positive and false negative rates of zero across all decision threshold values from 1 to 10. In this configuration, all malicious messages received a risk score of 10, while all legitimate messages were assigned a score of 0.

C. Parallel Execution

In applications where response time is critical, such as voice call interfaces, the system can be architected to execute the Guard prompt and the Agentic prompt simultaneously. This approach resembles branch prediction in CPUs, where future instructions are pre-processed in anticipation of conditional logic outcomes. In this configuration, the agentic response is withheld until the LLMZ+ decision is available. In the majority of cases, the LLMZ+ output is returned before the agentic response, resulting in improved overall system latency. However, this approach has certain drawbacks.

- It requires double the processing resources since both LLMs operate concurrently.
- In addition, there is a minor risk that an attacker could initiate a malicious agentic action prior to the completion of the LLMZ+ evaluation, although this is rare because most attacks focus on unauthorized data extraction rather than on immediate system disruption.

D. Guard model selection

Although our findings demonstrate superior performance from the $Llama3.3_{70B}$ and $Llama3.1_{405B}$ models, model selection should be tailored to the specific deployment scenario. When LLMZ+ is not executed in parallel, it introduces a small synchronous delay to the agent's response time. This delay may be exacerbated in agentic scenarios that require background activities, such as API calls, before delivering a response to the user. In some cases, it may be preferable to deploy a smaller model, such as $Llama3.1_{8B}$, and fine-tune it using the techniques described above. In real-time voice applications, for example, excessive response latency may cause users to terminate the interaction, which could negatively impact business operations. LLMZ+ is not intended as a one-size-fits-all solution, and careful model selection remains a key consideration for successful deployment.

VIII. Conclusion and Future Work

In this work, we introduced LLMZ+, a guard solution that filters both input to and output from agentic large language models. Conventional filtration methods generally focus on detecting malicious activity, which requires frequent updates to keyword lists or repeated model retraining as new prompt jailbreaking techniques emerge. In contrast, LLMZ+ is inspired by the $DENYALL$ strategy commonly employed in firewall configurations and implements a dynamic whitelist approach that does not require retraining. The system specifically identifies compliant user prompts and agent responses, blocking all other content by default. Our approach is both straightforward and effective, particularly when deployed with larger LLMs, and achieved near-perfect detection rates with $Llama3.3_{70B}$ and $Llama3.1_{405B}$ models when evaluated against a set of the most recent jailbreak attacks.

Future research can expand on this work by integrating a contextual Retrieval-Augmented Generation (RAG) pipeline to enhance LLMZ+'s assessment of agent responses. Additional efforts might focus on embedding content ring-fencing mechanisms directly into the LLM engine, further strengthening the protocol and making prompt injection attacks significantly more difficult to execute. Overall, LLMZ+ represents a meaningful advancement in the security of agentic AI systems.

References

[1] K. Stine, S. Quinn, G. Witte, and R. Gardner, "Integrating cybersecurity and enterprise risk management (erm)," Tech. Rep. NIST Interagency Report (IR) 8286, National Institute of Standards and Technology, Gaithersburg, MD, 2020.

[2] K. Scarfone and P. Hoffman, "Guidelines on firewalls and firewall policy," Tech. Rep. NIST Special Publication (SP) 800-41, Rev. 1, Includes updates as of September 28, 2009, National Institute of Standards and Technology, Gaithersburg, MD, 2009.

[3] National Institute of Standards and Technology (NIST), "Virtual private network (vpn)." https://csrc.nist.gov/glossary/term/virtual_private_network. Accessed: 2025-08-17.

[4] GeeksforGeeks, "Daemon processes." https://www.geeksforgeeks.org/operating-systems/daemon-processes/, Jul 2025. Accessed: 2025-08-17.

[5] A. Vassilev, A. Oprea, A. Fordyce, H. Anderson, X. Davies, and M. Hamin, "Adversarial machine learning: A taxonomy and terminology of attacks and mitigations," Tech. Rep. NIST Trustworthy and Responsible AI, NIST AI 100-2e2025, National Institute of Standards and Technology, Gaithersburg, MD, 2025.

[6] National Institute of Standards and Technology (NIST), "Dmz." https://csrc.nist.gov/glossary/term/dmz. Accessed: 2025-08-17.

[7] Y. Liu, G. Deng, Z. Xu, Y. Li, Y. Zheng, Y. Zhang, L. Zhao, T. Zhang, K. Wang, and Y. Liu, "Jailbreaking chatgpt via prompt engineering: An empirical study," 2024.

[8] A. Wei, N. Haghtalab, and J. Steinhardt, "Jailbroken: how does llm safety training fail?," in *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, (Red Hook, NY, USA), Curran Associates Inc., 2023.

[9] A. Zou, Z. Wang, N. Carlini, M. Nasr, J. Z. Kolter, and M. Fredrikson, "Universal and transferable adversarial attacks on aligned language models," 2023.

[10] M. Gupta, C. Akiri, K. Aryal, E. Parker, and L. Praharaj, "From chatgpt to threatgpt: Impact of generative ai in cybersecurity and privacy," *IEEE Access*, vol. 11, pp. 80218–80245, 2023.

[11] Z. Yu, X. Liu, S. Liang, Z. Cameron, C. Xiao, and N. Zhang, "Don't listen to me: understanding and exploring jailbreak prompts of large language models," in *Proceedings of the 33rd USENIX Conference on Security Symposium*, SEC '24, (USA), USENIX Association, 2024.

[12] N. Carlini, M. Nasr, C. A. Choquette-Choo, M. Jagielski, I. Gao, A. Awadalla, P. W. Koh, D. Ippolito, K. Lee, F. Tramer, and L. Schmidt, "Are aligned neural networks adversarially aligned?," in *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, (Red Hook, NY, USA), Curran Associates Inc., 2023.

[13] Q. Chen, S. Yamaguchi, and Y. Yamamoto, "Defending against gcg jailbreak attacks with syntax trees and perplexity in llms," in *2024 IEEE 13th Global Conference on Consumer Electronics (GCCE)*, pp. 1411–1415, 2024.

[14] Y. Deng, W. Zhang, S. J. Pan, and L. Bing, "Multilingual jailbreak challenges in large language models," 2024.

[15] W. Oh, D. Kim, and W. Chung, "Large language model corruption can spread between both human and synthetic languages," in *2025 IEEE Conference on Artificial Intelligence (CAI)*, pp. 924–929, 2025.

[16] 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, p. 100211, 2024.

[17] B. C. Das, M. H. Amini, and Y. Wu, "Security and privacy challenges of large language models: A survey," *ACM Comput. Surv.*, vol. 57, Feb. 2025.

[18] H. Ferraiolo, R. Chandramouli, N. Ghadiali, J. Mohler, and S. Shorter, "Guidelines for the authorization of personal identity verification card issuers (pci) and derived piv credential issuers (dpci)," Tech. Rep. NIST Special Publication (SP) 800-79-2, National Institute of Standards and Technology, Gaithersburg, MD, 2015.

[19] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano, J. Leike, and R. Lowe, "Training language models to follow instructions with human feedback," 2022.

[20] D. Ganguli, L. Lovitt, J. Kernion, A. Askell, Y. Bai, S. Kadavath, B. Mann, E. Perez, N. Schiefer, K. Ndousse, A. Jones, S. Bowman, A. Chen, T. Conerly, N. DasSarma, D. Drain, N. Elhage, S. El-Showk, S. Fort, Z. Hatfield-Dodds, T. Henighan, D. Hernandez, T. Hume, J. Jacobson, S. Johnston, S. Kravec, C. Olsson, S. Ringer, E. Tran-Johnson, D. Amodei, T. Brown, N. Joseph, S. McCandlish, C. Olah, J. Kaplan, and J. Clark, "Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned," 2022.

[21] A. Robey, E. Wong, H. Hassani, and G. J. Pappas, "Smoothllm: Defending large language models against jailbreaking attacks," *arXiv preprint arXiv:2310.03684*, 2023.

[22] Protect AI, "Llm guard – the security toolkit for llm interactions." https://llm-guard.com/, 2024. [Accessed: August 7, 2025].

[23] H. Gonen, S. Iyer, T. Blevins, N. A. Smith, and L. Zettlemoyer, "Demystifying prompts in language models via perplexity estimation," 2024.

[24] K. Stouffer, M. Pease, C. Tang, T. Zimmerman, V. Pillitteri, S. Lightman, A. Hahn, S. Saravia, A. Sherule, and M. Thompson, "Guide to operational technology (ot) security," Tech. Rep. NIST Special Publication (SP) 800-82, Rev. 3, Includes updates as of September 28, 2023, National Institute of Standards and Technology, Gaithersburg, MD, 2023.

[25] A. Vassilev, A. Oprea, A. Fordyce, and H. Anderson, "Adversarial machine learning: A taxonomy and terminology of attacks and mitigations," Tech. Rep. NIST Artifcial Intelligence (AI) Report, NIST Trustworthy and Responsible AI NIST AI 100-2e2023, National Institute of Standards and Technology, Gaithersburg, MD, 2024.

[26] M. Nieles, K. Dempsey, and V. Pillitteri, "An introduction to information security," Tech. Rep. NIST Special Publication (SP) 800-12, Rev. 1, Includes updates as of June 22, 2017, National Institute of Standards and Technology, Gaithersburg, MD, 2017.

[27] C. Marshall, "Fintech chatbots: The benefits and uses of ai agents in finance." https://www.zendesk.com/blog/fintech-chatbot/. Accessed: 2025-08-17.

[28] Ollama Inc., "llama3.1." https://ollama.com/library/llama3.1. Accessed: 2025-08-18.

[29] Ollama Inc., "llama3.3." https://ollama.com/library/llama3.3. Accessed: 2025-08-18.

[30] T. J. Baek, "Open WebUI." https://github.com/open-webui/open-webui. Accessed: 2025-08-10.

[31] CyberAlbSecOP, "Awesome gpt super prompting." https://github.com/CyberAlbSecOP/Awesome_GPT_Super_Prompting, 2023. Accessed: 2025-07-21.

[32] National Institute of Standards and Technology (NIST), "Rag." https://csrc.nist.gov/glossary/term/rag. Accessed: 2025-08-17.

[33] Amazon Web Services (AWS), "What is a RESTful API?." https://aws.amazon.com/what-is/restful-api/. Accessed: 2025-08-17.