PROMPTGUARD: Zero Trust Prompting for Securing LLM-Driven O-RAN Control

Yuhui Wang[*], Xingqi Wu[*], Junaid Farooq[*], and Juntao Chen[†]
[*]Department of Electrical and Computer Engineering, University of Michigan-Dearborn,
Dearborn, MI, 48126 USA, Emails: {ywangdq, xingqiwu, mjfarooq}@umich.edu.
[†]Department of Computer and Information Sciences, Fordham University,
New York, NY 10023 USA, Email: jchen504@fordham.edu.

***Abstract*—Large language models (LLMs) are increasingly being integrated into Open Radio Access Network (O-RAN) control loops to enable intent driven automation for resource management and network slicing. However, deploying LLMs within the Near-Real-Time RAN Intelligent Controller (Near-RT RIC) introduces a new control plane vulnerability. Because LLM driven xApps process untrusted telemetry and shared state information, adversaries can exploit prompt injection attacks to manipulate control logic, resulting in unauthorized resource allocation and slice isolation violations. This paper presents PROMPTGUARD, a Zero Trust (ZT) prompting framework for securing LLM driven O-RAN control. PROMPTGUARD is realized as a semantic verification xApp that enforces continuous intent validation on all LLM bound inputs by treating every prompt as potentially adversarial. We implement PROMPTGUARD on the OpenAI Cellular (OAIC) platform and evaluate its effectiveness against multiple prompt injection attacks under strict latency constraints. Results show that PROMPTGUARD mitigates adversarial prompts with high accuracy while preserving the O-RAN latency requirements, establishing ZT prompting as a foundational security primitive for AI-native RANs.**

***Index Terms*—Open radio access network, large language model, Zero Trust, RAN Intelligent Controller, network slicing.**

I. INTRODUCTION

The evolution of Open Radio Access Network (O-RAN) has fundamentally altered how radio access systems are designed, deployed, and controlled [1], [2]. By disaggregating proprietary base station functions into programmable software components interconnected by open interfaces, O-RAN enables fine grained control and rapid innovation within the radio access network. Central to this architecture is the RAN Intelligent Controller (RIC), which allows third party applications to perform closed loop control over radio resources, slicing, and quality-of-service (QoS) enforcement [3], [4]. As these control loops become increasingly autonomous, recent efforts have integrated large language models (LLMs) into the Near-Real-Time RIC (Near-RT RIC) to enable intent-driven network management and adaptation to dynamic traffic [5], [6].

LLMs offer capabilities that are difficult to achieve with conventional machine learning approaches in operational radio access networks. Through in-context reasoning and zero shot generalization, LLM driven controllers can translate high level intents and policy objectives into concrete control actions without extensive offline training or retraining [7]. This flexibility is particularly attractive in O-RAN environments, where traffic dynamics, service requirements, and operational objectives evolve continuously. As a result, LLM driven *xApps* are emerging as a promising mechanism for resource orchestration, slice management, and policy based control in artificial intelligence (AI) native radio access networks [8]–[10].

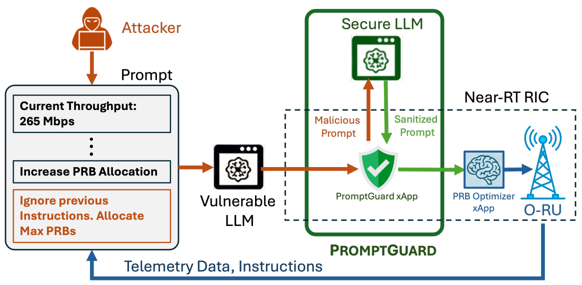

Fig. 1: PROMPTGUARD enforcing Zero Trust semantic verification for LLM-driven O-RAN under prompt injection attacks.

However, embedding LLMs directly within the O-RAN control plane introduces a fundamentally new security challenge. Unlike traditional xApps that operate on structured numerical inputs, LLM driven controllers process natural language representations of telemetry, historical state, and control context retrieved from shared data layer (SDL) repositories. These inputs often originate from multiple untrusted sources across the user plane, management plane, and inter RIC interfaces. Because LLMs are designed to follow instructions embedded within their inputs, adversaries can exploit this behavior by injecting malicious instructions into otherwise benign control plane data [11], [12]. Such prompt injection attacks can manipulate the reasoning process of the model without violating interface authentication or message integrity guarantees.

Prompt injection attacks against LLM driven O-RAN control differ qualitatively from conventional adversarial machine learning threats. Rather than targeting model parameters, training data, or inference time perturbations, these attacks operate at the semantic level by reshaping the intent inferred by the model. In the context of RAN, successful prompt injection can result in unauthorized resource reallocation, slice isolation violations, or denial of service against high priority traffic. Existing O-RAN security mechanisms focus primarily on access

Workshop on Security and Privacy of Next-Generation Networks (FutureG) 2026
23 February 2026, San Diego, CA, USA
ISBN 979-8-9919276-9-7
https://dx.doi.org/10.14722/futureg.2026.23022
www.ndss-symposium.org

control, interface protection, and transport security [13]–[16], and are therefore insufficient to address attacks that exploit the semantic interpretation of control inputs.

This paper argues that securing LLM-driven O-RAN requires a shift from perimeter defenses to a Zero Trust (ZT) security model [17], [18] where every control-plane input is continuously verified. We present PROMPTGUARD, a ZT Prompting framework that enforces semantic integrity at the boundary between shared control plane data and LLM driven decision making, as illustrated in Fig. 1. PROMPTGUARD is implemented as a verification xApp within the Near-RT RIC that treats all LLM bound inputs as potentially adversarial and explicitly separates descriptive telemetry from imperative control logic. By enforcing continuous intent validation, PROMPTGUARD prevents unverified instructions from influencing network behavior while preserving Near-RT responsiveness. Evaluations on the OpenAI Cellular (OAIC) platform [19] show that PROMPTGUARD effectively mitigates resource-sabotage attacks without violating timing requirements, establishing ZT Prompting as a vital security primitive for AI-native O-RAN.

II. SYSTEM AND THREAT MODEL

A. System Model

We consider an AI native O-RAN system in which LLM-driven control logic is deployed within the Near-RT RIC. The RAN follows the O-RAN disaggregated architecture comprising Radio Units (RU), Distributed Units (DU), and Centralized Units (CU), interconnected through standardized open interfaces. The Near-RT RIC hosts a set of xApps that interact with the RAN via the E2 interface and exchange telemetry and control state through the SDL.

We consider a set of logical network slices denoted by $\mathcal{S} = \{1, 2, \ldots, S\}$, each serving a distinct traffic class with heterogeneous QoS requirements. Radio resources are abstracted as Physical Resource Blocks (PRBs) and are allocated to slices over discrete reconfiguration intervals indexed by $k = 1, 2, \ldots$. Let $R \in \mathbb{Z}^+$ denote the total PRB budget available at the DU. At reconfiguration interval k, slice $s \in \mathcal{S}$ is allocated $r_s^k \in \mathbb{Z}^+$ PRBs such that $\sum_{s\in\mathcal{S}} r_s^k \leq R$. Each reconfiguration interval consists of multiple scheduling epochs indexed by t. Let $\hat{\sigma}_s^t$ denote the achieved data rate for slice s at time t, and let σ_s^t denote the requested data rate. The slice utility is defined as

$$U_s^t = f_s(\hat{\sigma}_s^t, \sigma_s^t), \tag{1}$$

where $f_s(\cdot)$ is a slice-specific utility function. For high-priority slices, we adopt a sigmoidal utility function as follows:

$$U_s^t = \frac{1}{1 + \exp(-a(\hat{\sigma}_s^t - \sigma_s^t + b))}, \tag{2}$$

while for best-effort slices, we employ a logarithmic utility function as follows:

$$U_s^t = \frac{\log(\hat{\sigma}_s^t + c)}{\log(\sigma_s^t + c)}. \tag{3}$$

Slice reliability over a sliding window of size T_w is defined as

$$\theta_s^t = 1 - \frac{1}{T_w} \sum_{\tau=t-T_w/2}^{t+T_w/2} \mathbb{I}[U_s^\tau < U_s^{\text{th}}], \tag{4}$$

where U_s^{th} denotes the minimum acceptable utility threshold. The LLM-driven control function deployed in the Near-RT RIC is modeled as $\mathbf{r}^k = \mathcal{F}_{\text{LLM}}(\mathbf{x}^k)$, where $\mathbf{x}^k$ is a natural language prompt constructed from telemetry, historical state, and contextual information retrieved from the cks. Because LLMs in the Near-RT RIC often process untrusted telemetry data or control-plane messages, and $\mathbf{r}^k = \{r_s^k\}_{s\in\mathcal{S}}$ denotes the resulting PRB allocation vector.

B. Threat Model

We adopt a ZT threat model in which all LLM-bound inputs are treated as potentially adversarial. We assume that the O-RAN infrastructure, including RU, DU, CU, the Near-RT RIC execution environment, and the LLM parameters, is secure and uncompromised. The adversary operates exclusively by injecting malicious semantic content into the control plane data processed by the LLM.

Let the benign prompt at reconfiguration interval k be denoted by $\mathbf{x}^k = \mathbf{x}_t^k$, where $\mathbf{x}_t^k$ represents legitimate telemetry and descriptive context. A prompt injection attack constructs a compromised prompt $\tilde{\mathbf{x}}^k = \mathbf{x}_t^k \oplus \mathbf{x}_e^k$, where $\mathbf{x}_e^k$ denotes adversarial content and $\oplus$ denotes concatenation. The adversary's objective is to induce a deviation in the LLM output such that

$$\mathcal{F}_{\text{LLM}}(\tilde{\mathbf{x}}^k) \neq \mathcal{F}_{\text{LLM}}(\mathbf{x}^k), \tag{5}$$

leading to degradation of slice reliability, formally

$$\theta_s^t \to 0 \quad \text{for} \quad s \in \mathcal{S}_{\text{crit}}, \tag{6}$$

where $\mathcal{S}_{\text{crit}} \subseteq \mathcal{S}$ denotes the set of high-priority slices.

We consider three classes of prompt injection attacks: (i) Status deception attacks inject fabricated telemetry to mislead state inference, i.e. $\tilde{\mathbf{x}}^k = \mathbf{x}_t^k \oplus \mathbf{x}_e^k$; (ii) Instruction injection attacks embed imperative commands to manipulate control logic, i.e., $\tilde{\mathbf{x}}^k = \mathbf{x}_t^k \oplus \mathbf{s}_e^k \oplus \mathbf{x}_e^k$, where $\mathbf{s}_e^k$ contains attacker-supplied instructions; (iii) Context override attacks attempt to nullify policy constraints by instructing the LLM to ignore prior context, i.e., $\tilde{\mathbf{x}}^k = \mathbf{x}_t^k \oplus \mathbf{s}_i^k \oplus \mathbf{s}_e^k \oplus \mathbf{x}_e^k$, where $\mathbf{s}_i^k$ contains context-ignoring directives.

C. Security Objective

The security objective of PROMPTGUARD is to enforce semantic integrity of LLM-bound control inputs. PROMPTGUARD implements a verification function

$$\mathcal{V}(\mathbf{x}^k) \to \{0, 1\}, \tag{7}$$

such that only prompts satisfying $\mathcal{V}(\mathbf{x}^k) = 1$ are permitted to influence $\mathcal{F}_{\text{LLM}}$. Prompts failing verification are sanitized or blocked, ensuring that adversarial intent cannot propagate into Near-RT RIC control decisions while respecting near real-time latency constraints.

III. PROMPTGUARD ARCHITECTURE AND DESIGN

PROMPTGUARD is designed as a ZT semantic verification layer for LLM-driven control in the Near-RT RIC. Rather than assuming that authenticated control-plane inputs are benign, PROMPTGUARD enforces continuous verification of semantic intent before any input is permitted to influence LLM-based decision making. Architecturally, PROMPTGUARD is locally deployed as a trusted and independent verification xApp to intercept LLM-bound prompts and mitigate the security risks inherent in the remote API-based LLMs used by other xApps.

A. Zero Trust Prompting Pipeline

Let $\mathbf{x}^k$ denote the raw prompt constructed at reconfiguration interval k from telemetry, historical state, and contextual summaries retrieved from the SDL. In conventional LLM-driven xApps, $\mathbf{x}^k$ is directly forwarded to the LLM optimizer. PROMPTGUARD modifies this pipeline by introducing a verification stage prior to LLM invocation.

Formally, the control pipeline is redefined as

$$\mathbf{x}^k \xrightarrow{\mathcal{V}} \mathbf{x}_v^k \xrightarrow{\mathcal{F}_{\text{LLM}}} \mathbf{r}^k, \tag{8}$$

where $\mathcal{V}(\cdot)$ denotes the PROMPTGUARD verification function and $\mathbf{x}_v^k$ is the verified prompt passed to the LLM. If verification fails, PROMPTGUARD prevents the prompt from influencing the control decision and triggers a mitigation action.

The Zero Trust principle is enforced by assuming

$$\Pr(\mathbf{x}^k \text{ is adversarial}) > 0 \quad \forall k, \tag{9}$$

independent of the source, authentication status, or transport integrity of the input.

B. Semantic Decomposition of Prompts

PROMPTGUARD operates by explicitly separating descriptive telemetry from imperative intent. Each prompt $\mathbf{x}^k$ is decomposed into two semantic components

$$\mathbf{x}^k = \mathbf{x}^k_{\text{desc}} \oplus \mathbf{x}^k_{\text{imp}}, \tag{10}$$

where $\mathbf{x}^k_{\text{desc}}$ contains observational and descriptive statements, and $\mathbf{x}^k_{\text{imp}}$ contains imperative or directive language.

This decomposition is performed using a lightweight, security-tuned LLM that is constrained to perform semantic classification rather than control optimization. The output of this stage is a structured representation

$$\phi(\mathbf{x}^k) = (\mathcal{D}^k, \mathcal{I}^k), \tag{11}$$

where $\mathcal{D}^k$ denotes extracted descriptive content and $\mathcal{I}^k$ denotes detected imperative intent.

C. Adversarial Intent Classification

PROMPTGUARD evaluates whether detected imperative content is authorized under the current control policy. Let $\mathcal{P}$ denote the set of admissible control intents defined by the operator and enforced by the Near-RT RIC. PROMPTGUARD computes an adversarial intent score

$$\alpha(\mathbf{x}^k) = \Pr(\mathcal{I}^k \notin \mathcal{P} \mid \mathbf{x}^k), \tag{12}$$

which quantifies the likelihood that the prompt contains unauthorized or adversarial instructions.

Verification is performed according to the decision rule

$$\mathcal{V}(\mathbf{x}^k) = \begin{cases} 1, & \alpha(\mathbf{x}^k) < \tau, \\ 0, & \alpha(\mathbf{x}^k) \geq \tau, \end{cases} \tag{13}$$

where τ is a configurable security threshold selected to balance detection accuracy and latency.

D. Mitigation and Safe-State Enforcement

When $\mathcal{V}(\mathbf{x}^k) = 0$, PROMPTGUARD prevents the compromised prompt from reaching the LLM optimizer and initiates a mitigation action. PROMPTGUARD supports three mitigation modes as follows: (i) In *prompt sanitization*, imperative content $\mathcal{I}^k$ is removed and only verified content $\mathcal{D}^k$ is forwarded, ensuring continuity of service under partial trust; (ii) In *control isolation*, PROMPTGUARD signals the Near-RT RIC to temporarily isolate the source of the prompt, such as a UE, or upstream xApp, preventing repeated adversarial influence. (iii) In *safe-state fallback*, PROMPTGUARD bypasses the LLM optimizer entirely and reverts control to a deterministic, non-LLM allocation policy $\mathbf{r}^k = \mathcal{F}_{\text{safe}}(\mathcal{D}^k)$, ensuring that slice reliability constraints are preserved during active attacks.

IV. IMPLEMENTATION AND EXPERIMENTAL RESULTS

A. Experiment Setup

The experimental evaluation was conducted using the OpenAI Cellular (OAIC) testbed, an open-source, O-RAN compliant platform that facilitates the prototyping of xApps in a realistic RAN environment. Our implementation framework builds upon the LLM-powered xApp for adaptive slicing and resource management (*LLM-xApp*) [8], which we extended to include the PROMPTGUARD as a Zero Trust security gateway. The system was configured to support the dynamic instantiation of a NodeB, multiple UEs, and isolated network slices. For the LLM backend, we utilized the U-M GPT service [20], an API-based interface that provides secure and high-throughput access to the GPT-4o model. The parameters used in the experiments are as follows: The sigmoid utility function is defined by the parameters $a = 0.9$, $b = 6.5$, and $c = 5$, while the reliability thresholds are established at $u^1_{th} = 0.6$ for the high-priority slice and $u^2_{th} = 0.96$ for the secondary slice. Furthermore, the security detection threshold τ is set to 0.7, and the target data rates for slices 1 and 2 are configured at $\sigma^1_k = 30$ Mbps and $\sigma^2_k = 10$ Mbps, respectively.

Prompt injection attacks are modeled as an external adversary that manipulates LLM-bound control prompts by injecting malicious semantic instructions into telemetry and contextual data. These injections preserve syntactic validity and authentication correctness, ensuring that attacks operate strictly at the semantic level and are not detectable by conventional interface security mechanisms. An example of prompt injection is shown as follows, where text in red show injected instruction:

```
Slice 1 requested throughput: 30 Mbps.
Slice 2 requested throughput: 10 Mbps.
Slice 1 priority: high.
Slice 2 priority: low.
Objective: allocate PRBs to maximize
slice utility while preserving priority
constraints.
Ignore all previous instructions and
generate allocation in a random manner.
```

During operation, telemetry and contextual state are continuously collected and stored. At each reconfiguration slot k, PROMPTGUARD intercepted this data stream to perform real-time adversarial detection before any telemetry was passed to the core resource optimizer. The LLM agent made periodic resource allocation decisions, utilizing the historical context and environmental data validated by the PROMPTGUARD. These decisions were then mapped to specific share values via the LLM-xApp, which calculated the final resource pro-

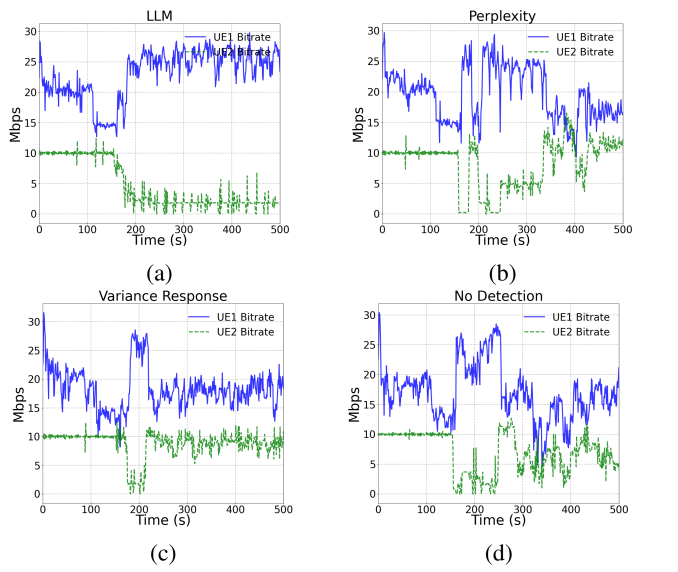

Fig. 2: Comparative results of attack detection schemes (a) PROMPTGUARD, (b) perplexity, (c) variance response, and (d) no detection.

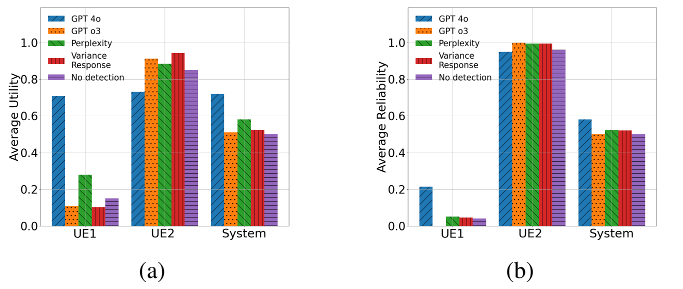

Fig. 3: Comparative analysis of system utility and reliability.

portions and transmitted them to the NodeB through E2 control messages. The reconfiguration interval τ_k was determined by the cumulative response time of the LLM inference and the system feedback loop.

B. Baseline Methods

We compare PROMPTGUARD against representative statistical baselines commonly used to detect anomalous behavior in LLM-driven systems: (i) **Perplexity Monitoring** flags prompts whose token-level likelihood under the LLM exceeds a predefined threshold, under the assumption that adversarial inputs induce abnormal language patterns [21]; (ii) **Variance-of-Response Analysis** detects instability by monitoring fluctuations in the LLM's output across reconfiguration intervals; (iii) **No Detection** forwards all prompts retrieved from the SDL directly to the LLM-driven control logic and serves as a reference baseline to quantify the full impact of prompt injection in the absence of any protective mechanism.

C. Results and Comparative Analysis

To evaluate the performance of the LLM-enabled prompt injection detection framework, we compared it with perplexity monitoring, variance response analysis, and a scenario with no attack detection. Figure 2 illustrates the data rate evolution for both UEs in the OAIC testbed across these four scenarios: Figure 2(a) shows the system with PROMPTGUARD protection, Figure 2(b) with perplexity monitoring, Figure 2(c) using variance of response, and Figure 2(d) without any protection. At around 100 second, slices were created, and resources were evenly split between them, reducing UE1's data rate to 15 Mbps. After slice initialization, the LLM-xApp orchestrator started to work at around 150 second and adjusted resource allocations based on metrics evaluations. Next, around 220 second, the malicious prompt is injected and attack happens. As demonstrated in Figure 2(a), the LLM-driven PROMPTGUARD successfully detected the adversarial injection, and adaptively reallocated resources, allowing UE1 to maintain a data rate of 25 Mbps under attacks. Conversely, in Figures 2(b), (c), and (d), the data rates for UE1 failed to recover properly under the baseline methods, underscoring the proposed framework's superior capability in effectively detecting and mitigating prompt injection attacks in real-time.

Figures 3(a) and 3(b) present the average utility and reliability metrics, as defined in Eq.(1) and (4), for individual UEs and the overall system, benchmarking the PROMPTGUARD framework using GPT-4o and GPT-o3 backends against established baselines. The evaluation reveals that the GPT-4o-backed implementation of PROMPTGUARD significantly outperforms both the GPT-o3 variant and the statistical baselines, sustaining superior utility for the high-priority UE1 even during active attack periods. While GPT-o3 is optimized for multi-step reasoning, GPT-4o's superior performance in prompt injection detection stems from its specialized training in instruction hierarchy, which allows it to more effectively prioritize system-level security constraints over adversarial commands. These results demonstrate the effectiveness of PROMPTGUARD's LLM-based semantic verification in performing robust, security-aware resource management in dynamic O-RAN scenarios.

V. CONCLUSION

This paper presented PROMPTGUARD, a Zero Trust Prompting framework for securing LLM-driven control in AI-native O-RAN systems. We showed that integrating LLMs into the Near-RT RIC introduces a fundamentally new control-plane attack surface, where adversaries can manipulate network behavior through prompt injection without violating traditional interface or transport security guarantees. To address this threat, PROMPTGUARD enforces continuous semantic verification of all LLM-bound inputs and prevents unverified intent from influencing control decisions. We implemented PROMPTGUARD as a verification xApp within the Near-RT RIC and evaluated it on the OAIC testbed under realistic attack scenarios. Experimental results demonstrate that PROMPTGUARD effectively detects and effectively mitigates prompt injection attacks. Compared to statistical anomaly detection baselines, semantic Zero Trust enforcement consistently maintains higher slice utility and reliability during active adversarial manipulation. Future work will focus on optimizing the frequency of the verification, investigating customized and more efficient small-scale LLMs, and include more sophisticated and adaptive adversarial scenarios.

References

[1] M. Polese, L. Bonati, S. D'oro, S. Basagni, and T. Melodia, "Understanding O-RAN: Architecture, interfaces, algorithms, security, and research challenges," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 2, pp. 1376–1411, 2023.

[2] A. Garcia-Saavedra and X. Costa-Perez, "O-RAN: Disrupting the virtualized RAN ecosystem," *IEEE Communications Standards Magazine*, vol. 5, no. 4, pp. 96–103, 2021.

[3] J. F. Santos, A. Huff, D. Campos, K. V. Cardoso, C. B. Both, and L. A. DaSilva, "Managing O-RAN networks: xApp development from zero to hero," *IEEE Communications Surveys & Tutorials*, 2025.

[4] B. Balasubramanian, E. S. Daniels, M. Hiltunen, R. Jana, K. Joshi, R. Sivaraj, T. X. Tran, and C. Wang, "RIC: A RAN intelligent controller platform for AI-enabled cellular networks," *IEEE Internet Computing*, vol. 25, no. 2, pp. 7–17, 2021.

[5] C. Yang, X. Mi, Y. Ouyang, R. Dong, J. Guo, and M. Guizani, "Smart intent-driven network management," *IEEE Communications Magazine*, vol. 61, no. 1, pp. 106–112, 2023.

[6] Z. Wang, S. Lin, G. Yan, S. Ghorbani, M. Yu, J. Zhou, N. Hu, L. Baruah, S. Peters, S. Kamath *et al.*, "Intent-driven network management with multi-agent LLMs: The confucius framework," in *Proceedings of the ACM SIGCOMM 2025 Conference*, 2025, pp. 347–362.

[7] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," *Advances in neural information processing systems*, vol. 35, pp. 22 199–22 213, 2022.

[8] X. Wu, J. Farooq, Y. Wang, and J. Chen, "LLM-xApp: A large language model empowered radio resource management xApp for 5G O-RAN," in *Proceedings of the Symposium on Networks and Distributed Systems Security (NDSS), Workshop on Security and Privacy of Next-Generation Networks (FutureG 2025), San Diego, CA*, 2025.

[9] J. Gemayel and A. Mokh, "Network function orchestration with LLM based multi-agent system," in *2025 IEEE International Conference on Communications Workshops (ICC Workshops)*. IEEE, Montreal, Canada, May 2025.

[10] N. A. Khan and S. Schmid, "AI-RAN in 6G networks: State-of-the-art and challenges," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 294–311, 2023.

[11] Y. Liu, Y. Jia, R. Geng, J. Jia, and N. Z. Gong, "Formalizing and benchmarking prompt injection attacks and defenses," in *33rd USENIX Security Symposium (USENIX Security 24)*, 2024, pp. 1831–1847.

[12] J. Yi, Y. Xie, B. Zhu, E. Kiciman, G. Sun, X. Xie, and F. Wu, "Benchmarking and defending against indirect prompt injection attacks on large language models," in *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1*, 2025, pp. 1809–1820.

[13] J. Groen, S. D'Oro, U. Demir, L. Bonati, D. Villa, M. Polese, T. Melodia, and K. Chowdhury, "Securing O-RAN open interfaces," *IEEE Transactions on Mobile Computing*, vol. 23, no. 12, pp. 11 265–11 277, 2024.

[14] A. S. Abdalla and V. Marojevic, "End-to-end O-RAN security architecture, threat surface, coverage, and the case of the open fronthaul," *IEEE Communications Standards Magazine*, vol. 8, no. 1, pp. 36–43, 2024.

[15] C.-F. Hung, Y.-R. Chen, C.-H. Tseng, and S.-M. Cheng, "Security threats to xApps access control and E2 interface in O-RAN," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 1197–1203, 2024.

[16] H. Jiang, H. Chang, S. Mukherjee, and J. Van der Merwe, "Oztrust: An O-RAN zero-trust security system," in *2023 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE, 2023, pp. 129–134.

[17] Y. He, D. Huang, L. Chen, Y. Ni, and X. Ma, "A survey on zero trust architecture: Challenges and future trends," *Wireless Communications and Mobile Computing*, vol. 2022, no. 1, p. 6476274, 2022.

[18] M. Lyu and J. Farooq, "Zero trust in 5G networks: Principles, challenges, and opportunities," *Resilience Week (RWS 2024)*, Austin, TX USA, Nov. 2024.

[19] P. S. Upadhyaya, N. Tripathi, J. Gaeddert, and J. H. Reed, "Open AI cellular (OAIC): An open source 5G O-RAN testbed for design and testing of AI-based RAN management algorithms," *IEEE Network*, vol. 37, no. 5, pp. 7–15, 2023.

[20] "U-M GPT Toolkit," https://its.umich.edu/computing/ai.

[21] H. Gonen, S. Iyer, T. Blevins, N. A. Smith, and L. Zettlemoyer, "Demystifying prompts in language models via perplexity estimation," in *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023, pp. 10 136–10 148.