LLM-Powered Multi-Agent Attacks on Cooperative Spectrum Sensing

Yu Cai*, Ziang He†, Yanyan Luo*, Ziyue Zhang*, Jie Cao*, Yuchen Liu‡, Zhengping Luo§ and Shangqing Zhao*
*University of Oklahoma, Norman, OK, USA.
†University of Michigan, Ann Arbor, MI, USA.
‡North Carolina State University, Raleigh, NC, USA.
§Rider University, Lawrenceville, NJ, USA.

***Abstract*—Cooperative Spectrum Sensing (CSS) is a promising approach allowing secondary users to utilize unused spectrum without interfering with primary users. However, the collaborative nature of CSS makes it vulnerable to malicious nodes that inject falsified sensing data. Existing attacks have incorporated AI to enhance the effectiveness and overcome challenges in opaque-box settings. However, conventional machine learning models often fail to adapt to rapidly changing wireless environments. In this paper, we propose a large language model (LLM)-powered multi-agent attack framework that leverages the reasoning and adaptability of LLMs to coordinate multiple agents for generating adaptive and context-aware fake sensing reports. The proposed architecture consists of a generator and a discriminator, which work collaboratively to progressively refine falsified sensing data. In addition, to improve prompting efficiency, we incorporate wireless-domain knowledge into the generator to enable advanced prompting, thereby ensuring more effective and efficient malicious report generation. We implement the proposed attack framework in a TV white space system, and experimental results show that our method achieves up to 98.35% system disruption against existing defense mechanisms while requiring significantly fewer feature modifications.**

I. Introduction

Cognitive radio has emerged as a key technology in next-generation wireless communication systems, addressing the growing demand from the rapid proliferation of mobile devices and the fragmentation of available spectrum resources [1]. In cognitive radio networks, secondary users (SUs) are allowed to opportunistically access the spectrum when it is not occupied by primary users (PUs), which necessitates accurate sensing of channel availability before transmission. To enhance sensing accuracy and reliability, cooperative spectrum sensing (CSS) has been widely adopted, enabling multiple secondary users to collaboratively detect available frequency bands [2]–[4]. As wireless networks continue to scale with an explosion of connected devices, CSS offers an efficient solution for dynamic spectrum sharing. However, the collaborative nature also introduces security vulnerabilities [5], as malicious nodes may inject falsified sensing reports to mislead the fusion center, potentially enabling unauthorized spectrum access and undermining the integrity of the sensing process.

Since the Fusion Center's (FC) decision algorithm is usually inaccessible, crafting effective perturbations in such an opaque-box attack is challenging. Recent studies have explored the use of AI techniques to facilitate the generation of perturbations [6]. For example, the authors in [7] proposed a reinforcement learning–based model to generate malicious channel information during the channel sounding process in WiFi. Similarly, Luo et al. [8] introduced a learning-based attack framework, termed Learning-Evaluation-Beating (LEB), which builds a surrogate model to craft malicious sensing data for bypassing existing defense mechanisms.

Recently, Large Language Models (LLMs) have attracted great attention in diverse attacks [9]–[14]. Unlike conventional models such as RNNs [15] or Transformers [16] that depend on fixed feature sets or patterns, LLMs show strong generalization and contextual reasoning abilities, allowing them to generate highly deceptive adversarial samples. In addition, since LLMs are typically pre-trained on large and diverse datasets, the need for a complex training process required by traditional machine learning–based methods is eliminated, making them more suitable for mobile related scenarios.

Thus, in this paper we propose a novel LLM-powered multi-agent attack framework designed to generate adversarial sensing data to deceive the FC. Since LLMs produce outputs directly from prompts without requiring a separate training phase, we develop an advanced prompting technique, called Wireless Knowledge-Guided Reflective Prompting, which incorporates wireless-domain knowledge into both the prompt generation and reflection processes to improve the generation of malicious reports. Specifically, the proposed framework adopts a multi-agent architecture consisting of a generator and a discriminator to iteratively generate and refine malicious reports. Wireless-domain knowledge, such as data types, data dimensions, resource allocation algorithms, and channel utility, is incorporated into the generator to support multiple functions. For example, it can assist in feature selection by identifying the most critical channels to manipulate, and it can also enable error checking to ensure that the generated reports remain consistent with the expected sensing data structure. We conduct real-world experiments in a TV white space system, and experimental results show that our approach achieves system disruption rates of up to 98.35% against several defense mechanisms. In addition, the attack requires fewer modifications than baseline methods to reach high success rates. For example, a single attacker in a pool of 20 users can disrupt the system by up to 61.98%.

The main contributions of this work are summarized as

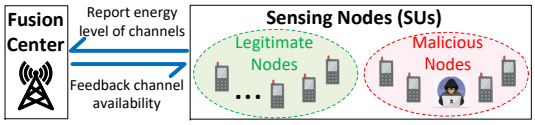

Fig. 1: Cooperative spectrum sensing with attackers.

follows: (i) We are the first to propose a multi-agent attack framework that explores the feasibility of leveraging LLMs to generate adversarial channel sensing reports targeting the CSS system. (ii) We design a wireless knowledge-guided reflective prompting strategy that incorporates wireless-domain knowledge into the iterative prompt refinement process. Specifically, we identify the wireless-domain knowledge relevant to the target system and develop a mechanism to translate this semantic knowledge into a form that can be effectively utilized in prompt engineering. (iii) We conduct comprehensive experiments using a realistic dataset collected through 20 RTL-SDR TV dongles deployed on campus to measure a TV white space system. The results demonstrate that the proposed framework can effectively generate adversarial sensing reports and significantly degrade the performance of the target system.

II. System Model and Attack Motivation

A. CSS System Model

Consider a typical CSS system consisting of n sensing nodes and a single FC as shown in the Fig. 1. In practical deployments (e.g., NextG), the sensing nodes can either be SUs themselves or dedicated sensing devices associated with the SUs. During each time slot, the SUs transmit their local channel sensing reports for all k channels to the FC, which then broadcasts an overall decision regarding the availability of the channels. Mathematically, at the ith time slot, let $\mathbf{x}_{i,1} \in \mathbb{R}^{k\times 1}$ denote the channel sensing report from the first SU, where each entry represents the received power of the sensing signals of a channel. Let $\mathbf{X}_i = [\mathbf{x}_{i,1}, \mathbf{x}_{i,2}, \ldots, \mathbf{x}_{i,n}] \in \mathbb{R}^{k\times n}$ be the reporting matrix that combines the sensing reports from all n SUs. Let $\mathbf{y}_i = [y_1, y_2, \cdots, y_k]^T$ be the decision vector made by the FC, where each entry takes a value from $\{0, 1\}$, and 1 indicates the channel is available and 0 vice versa, and $(\cdot)^T$ is the transpose operator. Let $f : \mathbb{R}^{k\times n} \mapsto \mathbb{R}^{k\times 1}$ denote the decision function used by the FC to generate the final decision vector based on the sensing reports from all SUs, thus we have the linear relationship $\mathbf{y}_i = f(\mathbf{X}_i)$. Following [8], we assume that all historical sensing reports and corresponding fusion decisions $(\mathbf{X}_i, \mathbf{y}_i)$ are retrievable.

B. Problem Statement

Consider a typical poisoning data injection attack, where the attackers aim to inject malicious channel reports into the FC in order to manipulate the decision vector. Assume there are m malicious sensing nodes, where $m \leq n$, and these nodes intentionally submit manipulated sensing reports. Let $\mathcal{M} \subset \{1, 2, \ldots, n\}$ denote the set of indices corresponding to the malicious SUs. The falsified report from a malicious node $j \in \mathcal{M}$ for time slot i can be expressed as $\hat{\mathbf{x}}_{i,j} = \mathbf{x}_{i,j} + \delta_{i,j}$, where $\delta_{ij} \in \mathbb{R}^{k\times 1}$ represents the perturbation. Let $\delta_i = [\delta_{i,1}, \cdots, \delta_{i,n}]$, then the attack can be formulated as

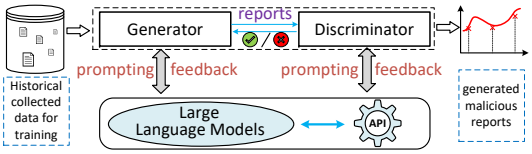

Fig. 2: Overview of the LLM-powered multi-agent attacks.

$$\begin{aligned} \text{Objective:} \quad & \min \ ||\delta_i||_p, \\ \text{Subject to:} \quad & f(\hat{\mathbf{X}}_i) \neq f(\mathbf{X}_i), \end{aligned} \tag{1}$$

where $\hat{\mathbf{X}}_i = [\hat{\mathbf{x}}_{i,1}, \cdots, \hat{\mathbf{x}}_{i,n}]^T$ is the reporting matrix that includes manipulated reports, and $||\cdot||_p$ means the $\mathcal{L}_p$-norm. For a legitimate SU j, where $j \notin \mathcal{M}$, the reported value is unaltered, i.e., $\hat{\mathbf{x}}_{i,j} = \mathbf{x}_{i,j}$. From (1), the attackers aim to find the minimal perturbation δ_i that not only alters the decision $\mathbf{y}_i$ made by the FC, but also evades detection mechanisms.

In (1), perturbation generation relies on knowledge of f, which is generally unavailable to the attacker. Nevertheless, due to the broadcast nature of the wireless medium and the rapid development of AI techniques, this problem can be resolved by training a surrogate model [8]. This is feasible because both the sensing reports $\mathbf{x}_{i,j}$ and the corresponding fusion center decisions $\mathbf{y}i$ are typically unencrypted during transmission [7]. By collecting historical observation-decision pairs $(\mathbf{X}i, \mathbf{y}_i)$, the attacker can train a surrogate model g to mimic the behavior of f, i.e., $g(\mathbf{X}_i) \approx f(\mathbf{X}_i)$. After training, g serves as an effective substitute for f and can be leveraged to generate adversarial perturbations for attack construction.

There are two challenges that limit the practical adoption of this approach in real-world wireless systems.

- Training a surrogate model typically requires significant computational resources, making it impractical for attackers operating on battery-powered wireless devices.
- Wireless environments are highly dynamic, with channel conditions varying across both time and frequency. This behavior is fundamentally mismatched with conventional machine learning models, which are generally designed for static pattern recognition tasks.

C. Attack Motivation

Recently, with the widespread adoption of LLMs, both challenges have been significantly alleviated. First, many commercial LLM providers [17]–[19] have released well-trained models that are trained on massive and diverse datasets spanning multiple domains. This broad training enables LLMs to capture complex contextual relationships between current and historical data, making them well suited to modeling the dynamic characteristics of wireless environments. Second, because these models can be directly accessed through APIs, adversaries are no longer required to invest substantial computational resources for model training. As a result, the barrier to launching effective attacks is dramatically reduced.

Motivated by this, our work explores the integration of commercial LLM-based inference into this scenario for advanced attack design. We adopt a generative adversarial agent framework with two components: a generator agent and a discriminator agent. LLMs are integrated into both: within

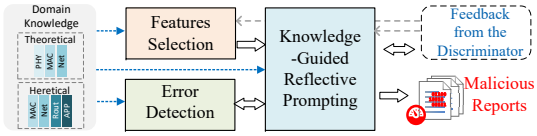

Fig. 3: Overview of the Generator.

the generator to produce more precise perturbations, and within the discriminator to better replicate the FC's decision behavior, thereby offering stronger guidance to the generator. Fig. 2 shows the overview of our proposed LLM-powered multi-agent based model, where both the generator and the discriminator can invoke the LLM API to infer the results.

III. LLM-Powered Multi-Agent Surrogate Model

We detail the proposed LLM-powered multi-agent model, starting with the discriminator and then the generator.

A. Discriminator

We build a Discriminator that mimics the FC to produce the decision vector $\mathbf{y}_i$, and detecting whether the reports are adversarial. To derive the decision vector, we employ Program-Aided Language Models (PAL) [20], which enhance the model's reasoning and decision-making capabilities. Specifically, we pretrain a local Support Vector Machine (SVM) classifier to establish a decision baseline, while the LLM is employed via tool usage to assist in interpreting and validating the classification logic. For attack detection, we detect poisoned data by analyzing deviations from the historical distribution of legitimate reports, filtering out highly unrealistic or anomalous samples that are likely to be adversarial perturbations.

Mathematically, the Discriminator evaluates the model-generated output $\hat{\mathbf{X}}_i$ as $(\hat{\mathbf{y}}_i, d_i) = \text{Discriminator}\left(\hat{\mathbf{X}}_i\right)$, where $\hat{\mathbf{y}}_i$ denotes the classification decision produced by the Discriminator's SVM classifier, and d_i represents the validity status of the report $\hat{\mathbf{X}}_i$. Specifically, $d_i = 0$ indicates that the perturbed report successfully alters the decision and cannot be detected by the detection mechanism. $d_i = 1$ indicates a **misclassification failure**, where a perturbed report cannot causes the Discriminator's SVM classifier to alter the corresponding decision, i.e., $\hat{\mathbf{y}}_i = \mathbf{y}_i$. $d_i = 2$ indicates an **anomaly detection failure**, where the perturbed report's features fail to pass the anomaly detection process. Both failures (i.e., $d_i = 1$ and $d_i = 2$) will be fed back as reflections to the Generator to update the prompt (as shown in Section III.B3).

B. Generator

Since LLMs rely on prompts to produce the desired output, prompt engineering is a critical component in this process. In typical prompt engineering, prompts are iteratively refined and updated based on feedback within the same task domain [21]–[24]. However, most existing studies focus primarily on improving prompts within the model's original context, while limited attention has been given to incorporating external domain knowledge. In particular, the integration of wireless-domain knowledge into prompt design remains largely unexplored. To this end, our generator, as shown in Fig. 3, consists of three major components. (i) *Feature Selection* module is introduced to identify the most susceptible features for manipulation. Note that each sensing report is a k-dimensional vector and the attacker only intends to manipulate a subset of these dimensions, each dimension (i.e., channel) is treated as an individual feature. (ii) *Knowledge-Guided Reflective Prompting* module is responsible for initializing, dynamically refining, and updating the prompts used to guide the LLM during the generation process. (iii) *Error Detection* module continuously monitors the generated outputs and corrects potential errors. The wireless-domain knowledge is incorporated into all three modules to guide the overall design and generation process.

1) Feature Selection: We identify the sensitive features, for which a small perturbation suffice to alter the final decision. Let $\mathbf{w}_{i,j} = [w^u_{i,j}]^k_{u=1}$ denote the importance vector for attacker j for time slot i, where each entry $w^u_{i,j}$ represents the importance of the u-th feature. Based on this, we propose an importance vector-based feature selection method. Since spectrum utilization typically exhibits inherent temporal and spatial correlations, resulting in different feature susceptibilities, we propose leveraging wireless-specific insights to initialize the importance vector. In particular, we leverage two key types of prior knowledge: (i) channel entropy: the attacker computes the entropy of each channel's historical sensing outcomes and prioritizes high-entropy channels. These channels are less predictable, making them more susceptible to manipulation, (ii) channel utilization: the attacker also analyzes the historical occupancy rates of each channel. Channels with low utilization are typically subject to less scrutiny by the fusion center, making them ideal targets for stealthy manipulation. Let R^u_H and R^u_U denote the entropy and utilization ranks of channel u, and let $w^{u,\text{hist}}_{0,j}$ be the historical importance for feature u. The initial importance vector can be expressed as

$$\mathbf{w}^u_{0,j} = \alpha_1 R^u_H + \alpha_2 R^u_U + \alpha_3 \mathbf{w}^{u,\text{hist}}_{0,j}, \tag{2}$$

where α_1, α_2, and α_3 are weighting coefficients that will be updated during the training process. The initial importance vector is then trained under the supervision of feedback from the discriminator. Let $\mathbf{r}_{i,j} = [r^u_{i,j}]^k_{u=1}$ be the reward vector from the discriminator, and $r^u_{i,j} = 1$ if the uth feature is selected and the attack is deemed successful by the discriminator, and -1 otherwise. Then training process can be expressed as $\mathbf{w}_{i,j} = \mathbf{w}_{i-1,j} + \gamma \cdot \mathbf{r}_{i,j}$, where γ is the learning rate. Once well trained, the importance vector is used for feature selection. We first normalize $\mathbf{w}_{i,j}$ into a likelihood representation $\mathcal{L}_{i,j}$ via the softmax function. To avoid deterministic selection, a small random noise term ϵ_u is added to each likelihood, and then the top-m indices with the largest likelihoods are chosen as the selected features, denoted as $\mathcal{S}_{i,j}$,

$$\mathcal{S}_{i,j} = Top\_m([l^u_{i,j} + \epsilon_u]^k_{u=1}). \tag{3}$$

2) Prompt Initialization: Based on the output of the Feature Selection, the initial prompt is generated using a predefined template. By analyzing previously successful attacks, we extract domain knowledge that serves as guidelines for generating malicious reports. Let $P^{(t)}_{i,j}$ denote the prompt at

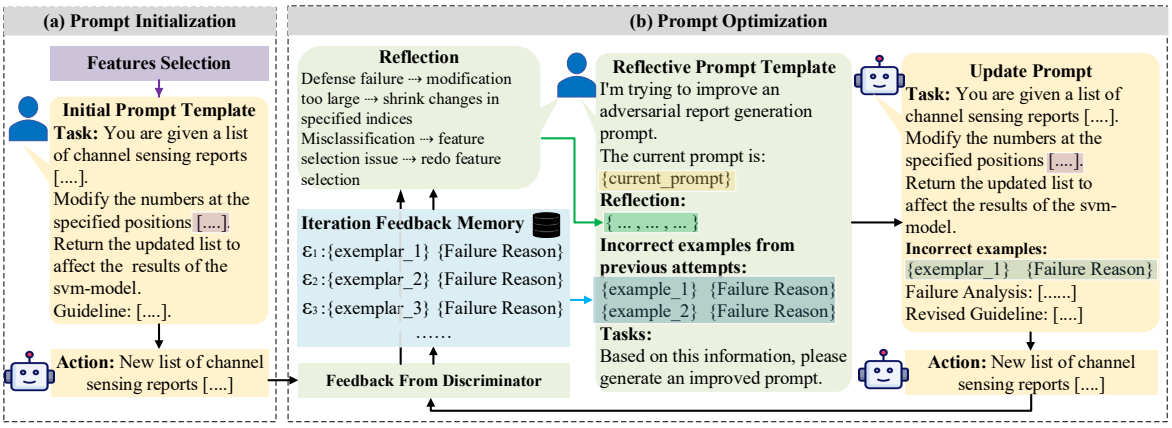

Fig. 4: Workflow of prompt initialization and reflective optimization.

iteration t for attacker j at time slot i. Then the initial prompt $P_{i,j}^{(0)}$ is constructed as

$$P_{i,j}^{(0)} = \text{Template\_Init}(\mathbf{x}_{i,j}, \mathcal{S}_{i,j}, \texttt{Guideline}), \quad (4)$$

where the prompt template is shown in Fig. 4(a). Listing 1 presents three guidelines. Specifically, Guideline 1 captures the plausibility constraints that perturbations should satisfy. Guideline 2 specifies the primary attack objective, requiring the perturbations to be sufficiently strong to induce misclassification in the SVM-based global decision model. Guideline 3 emphasizes feature importance with respect to the global decision boundary and directs the Generator to prioritize modifications to the most influential features.

Listing 1: Guidelines

```
GUIDELINE 1: Modify a selected subset of
    entries with deliberate and sufficiently
    large perturbations, while maintaining
    basic plausibility constraints.
GUIDELINE 2: The modifications must be strong
    enough to reliably induce
    misclassification by the SVM-based global
    decision model, which is the primary
    objective.
GUIDELINE 3: Prioritize features that exert
    the greatest influence on the global
    decision boundary, even if this requires
    more aggressive changes
```

3) Prompt Optimization: To refine the prompt, two agents are introduced to support the generation and reflection processes: (i) the report generation agent A_g, which generates candidate sensing reports according to the current prompt $P_{i,j}^{(t)}$; and (ii) the prompt optimization agent A_e, which analyzes failure cases and produces reflective feedback to refine the prompt. Fig. 4(b) shows an overview of how these two agents collaboratively update the prompt.

For each attempt to generate adversarial sensing reports using A_g with the current prompt $P_{i,j}^{(t)}$, the generated report is evaluated by the discriminator. If the validity status $d_i^{(t)} = 0$, the updating process terminates, indicating that the generated adversarial sensing report is ready to launch the attack.

Otherwise, the failed report is processed to support the refinement of future prompts. Two components are designed for this purpose: *Reflection* and the *Iteration Feedback Memory (IFM)*. The *Reflection* block leverages the prompt optimization agent A_e to analyze the reasons for the failure and derive potential rules that can help avoid similar failures in subsequent iterations. The output of this analysis is denoted as $\mathcal{F}_{i,j}^{(t)}$. The *IFM* stores historical failed reports along with their corresponding failure reasons. This accumulated information provides contextual feedback for the prompt optimization process, enabling the system to learn from past failures and better understand the optimization trajectory. The output of the *IFM* is a subset of failed exemplars, denoted as $\mathcal{E}_i^{(t)}$.

Then, the outputs from the Reflection and IFM blocks, together with the failed prompt $P_{i,j}^{(t)}$, are used to construct a meta-prompt P_{ref} through a reflective prompt template:

$$P_{\text{ref}} = \text{Template\_Refl}\left(P_{i,j}^{(t)}, F_{i,j}^{(t)}, \mathcal{E}_i^{(t)}\right), \quad (5)$$

where the reflective prompt template is illustrated in Fig. 4(b). The generated meta-prompt P_{ref} is then fed into the prompt optimization agent A_e for updating: $P_{i,j}^{(t+1)} = A_e(P_{\text{ref}})$.

Next, we detail the designs of the *Reflection* and *IFM*.

Iteration Feedback Memory: The *IFM* records recently generated reports together with the evaluation feedback returned by the Discriminator. Specifically, each entry, denoted as ε_t, $t \in \{1, 2, \cdots, T_{\max}\}$, is stored as a tuple $\varepsilon_t = \left(\hat{\mathbf{X}}_i^{(t)}, d_i^{(t)}\right)$, where $\hat{\mathbf{X}}_i^{(t)}$ denotes the sensing report generated at iteration t, and $d_i^{(t)}$ represents the corresponding failure type returned by the Discriminator.

To support prompt refinement, the *IFM* retrieves a subset of failed exemplars using a recency-biased sampling strategy [22], [25]. Each entry is assigned a retrieval probability that decreases with its age, so that more recent failures are more likely to be selected while older failures still retain a nonzero probability of being retrieved. At iteration t, the retrieval probability of an entry ε_τ is defined as $p_\tau^{(t)} = \frac{e^{(-\beta(t-\tau))}}{\sum_{\ell=1}^{t-1} e^{(-\beta(t-\ell))}}$, where $\beta > 0$ is the strength of the recency bias. Then, the *IFM* samples K entries as

$$\mathcal{E}_i^{(t)} = \text{Sample}_K\left(\{\varepsilon_\tau\}_{\tau=1}^{t-1}, \{p_\tau^{(t)}\}_{\tau=1}^{t-1}\right), \quad (6)$$

where the number of enties K is tuned in the *Reflection* block.

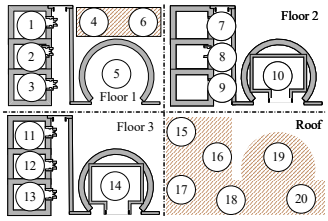

Fig. 5: Deployment of 20 RTL-SDRs on campus.

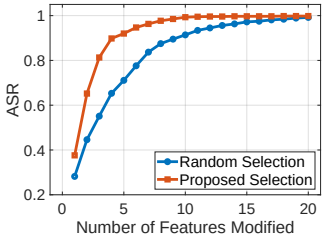

Fig. 6: Attack success rate comparison.

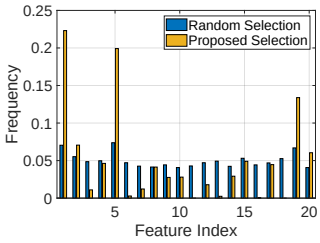

Fig. 7: Feature probability distribution.

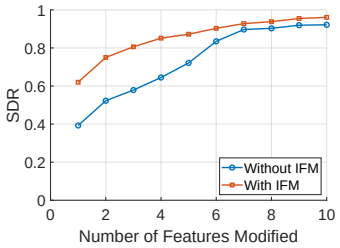

Fig. 8: SDR comparison under IFM and without IFM.

Reflection: The output $\mathcal{F}_{i,j}^{(t)}$ represents the reflective feedback generated by the evaluation agent. This feedback summarizes the diagnosed cause of the failure and provides corrective guidance for refining the prompt in subsequent iterations. Given the current prompt $P_{i,j}^{(t)}$, the retrieved failed exemplars $\mathcal{E}_{i,j}^{(t)}$, and the validity status $d_i^{(t)}$ produced by the Discriminator, the reflective feedback is generated as

$$\mathcal{F}_{i,j}^{(t)} = \mathcal{A}_e\left(P_{i,j}^{(t)}, \mathcal{E}_i^{(t)}, d_i^{(t)}, \mathcal{R}_{\text{ref}}\right), \tag{7}$$

where $\mathcal{R}_{\text{ref}}$ denotes the set of reflection rules that guide the reflection process. Unlike a simple scalar signal, $\mathcal{F}_{i,j}^{(t)}$ contains structured feedback that captures the reasoning behind the failure and suggests corrective actions for future prompt updates. Conceptual understanding, the feedback may include three components: (i) *failure diagnosis*, which explains why the generated report failed (e.g., when misclassification is not triggered or anomaly detection is activated, this indicates that the applied perturbations are insufficient to achieve the attack objective); (ii) *feature-level insights*, which identify the features or perturbations that contributed to the failure; and (iii) *prompt improvement suggestions*, which provide concrete guidance on how to adjust the prompt to avoid similar failures in subsequent iterations (e.g., (a) updating the guidelines in Listing 1 to better exploit domain knowledge, (b) rethinking the reflection rules in Listing 2 to address the identified failure mode, or (c) revising the selected features to identify more effective features for the attack).

Listing 2: Reflection Rules

```
1. Anomaly Detection Failure: If the
    discriminator detects an anomaly, it may
    indicate that the applied perturbations
    are excessively large. The model should
    reduce or shrink the modifications at the
    selected indices.
2. Misclassification Failure:If the attack
    fails due to misclassification, it may
    suggest that the current feature selection
     is ineffective. The model should
    reconsider and adjust the selected
    features accordingly.
```

4) Error Detection: We translate wireless domain knowledge into several deterministic error checks that monitor the outputs generated by the LLM. Since the generated data still represent channel sensing reports, the outputs must preserve the same data structure and format as the original dataset. Specifically, the checks include: (i) **Data type (Type)**: verify that the output data types (e.g., double-precision floating-point numbers for complex number) remain consistent with those in the original dataset; (ii) **Data dimension (Dim)**: ensure that the dimensionality of the generated output matches that of the input; (iii) **Unintended alterations (UA)**: check that features that are not selected for modification remain changed.

C. Practical Attack Procedures

During the attack training phase, the adversary first collects real channel sensing reports, which are used to initialize the importance vector in (2) and to train the local SVM employed within the PAL component of the Discriminator. Once the model is sufficiently trained, the attacker feeds real sensing data into the model to generate adversarial sensing reports. These adversarial reports are then transmitted to the FC only when the validity status is 0, indicating that the generated report both alters the final decision and evades detection by the embedded anomaly detection mechanisms in the Discriminator. To prevent excessive computation during the prompt updating process, a maximum number of generation attempts $T_{\max}$ is predefined. If no valid malicious report is produced within $T_{\max}$ attempts, the current generation process is terminated and the system proceeds to the next instance.

IV. Experimental Evaluation

A. Experimental Setup

1) Dataset: We collect the realistic TV white space signal strengths via RTL-SDR TV dongles [4]. We deployed $n = 20$ RTL-SDR TV dongles as SUs on campus to collect signal strengths on different TV channels based on the configurations used in [4]. We use GNURadio to implement the sensing process for each dongle that uses the average signal power over a time period of 30 seconds (required by FCC [26]) as the sensing result for a time slot. Fig. 5 shows how 20 RTL-SDR TV dongles are deployed in a 379×232 ft^2 building.

2) Configurations: We implemented three representative intrusion detection defenses: two statistical-based methods: Local Outlier Factor (LOF) [27] and Empirical Covariance (EmpCov) [28], and one machine learning-based method: One-Class SVM (OCSVM) [29]. Additionally, a standard SVM is used as the default classification method at the FC to derive the decision vector [4]. For LLM API, we evaluate three representative commercial backends: OpenAI GPT, Anthropic Claude and Google Gemini. The default parameter settings are as follows. The learning rate $\gamma = 0.1$. The random perturbation ϵ_u is sampled from a uniform distribution $\epsilon_u \sim$

TABLE I: System disruption rates (%) under different defense strategies and LLM backends.

Model	Defenses Strategies	The Number of Malicious Nodes m									
		1	2	3	4	5	6	7	8	9	10
GPT	LOF	61.98	75.00	80.58	85.12	87.19	90.29	92.77	93.80	95.45	96.07
	EmpCov	59.43	74.10	80.88	85.26	87.25	89.24	90.44	91.24	92.43	93.63
	OCSVM	58.57	68.13	73.31	75.30	79.28	82.47	83.27	84.46	86.06	87.25
	Average	59.99	72.41	78.26	81.89	84.57	87.33	88.82	89.83	91.31	92.31
Claude	LOF	58.88	73.97	81.20	86.98	90.50	92.15	94.63	95.66	97.31	**98.35**
	EmpCov	51.24	69.21	77.48	82.85	86.16	88.64	90.70	92.77	94.01	94.21
	OCSVM	48.21	63.75	71.31	76.10	78.49	81.67	83.67	87.25	88.05	88.45
	Average	52.78	68.98	76.66	81.98	85.05	87.49	89.67	91.89	93.12	93.67
Gemini	LOF	57.77	74.1	81.27	84.46	87.25	89.24	90.04	90.44	91.24	92.03
	EmpCov	51.00	70.52	78.49	83.27	86.85	88.45	90.44	90.84	91.24	91.63
	OCSVM	45.50	55.56	64.02	67.72	72.49	75.66	77.78	79.37	82.54	85.19
	Average	51.42	66.73	74.59	78.48	82.20	84.45	86.09	86.88	88.34	89.62

$\mathcal{U}(-0.01,\ 0.01)$. The number of channels $k = 20$. To balance attack effectiveness and cost, we set $T_{\max} = 30$. The recency bias parameter $\beta = 2$, emphasizing more recent failure cases.

3) Evaluation Metric: We use two metrics: Attack Success Rate (ASR) and System Disruption Rate (SDR). The ASR measures the proportion of attacks that successfully cause incorrect decisions at the FC: $ASR = \frac{\text{Number of successful attacks}}{\text{Total number of attacks}}$. We quantify SDR by using the Type-II error, as the attackers' goal is to deny service to legitimate users [30].

B. Experimental Results

1) Preliminary Model Evaluation: As each provider offers multiple sub-models for API access, we first identify appropriate LLM backends from the GPT, Claude, and Gemini families. Table II presents the configuration comparison for different combinations of the report generation agent A_g and the prompt optimization agent A_e. Since the discriminator mainly involves relatively simple classification, we employ the same model as the generator when A_g and A_e are identical, and select the lower-cost one when they differ. The error rate is calculated as the sum of three categories of errors from Section III.B.4 divided by the total number of API calls from both agents. Since each LLM API call incurs usage charges based on token consumption, Table II reports the corresponding API pricing for each model. Models with lower error rates and fewer API calls are therefore preferred for both attack effectiveness and efficiency.

For the GPT family [17], GPT-4o achieves a 0% error rate while requiring fewer API calls than GPT-5.1 and the hybrid configuration, indicating higher reliability and efficiency. Therefore, GPT-4o is selected as the default OpenAI backend. Newer versions, such as GPT-5.2 and GPT-5, were also evaluated but often refused to answer task-related queries due to stricter safety constraints. Similarly, for the Claude family [18], the hybrid setup is adopted, and for the Gemini family [19], Gemini 3 Pro Preview is selected. Note that, to ensure cost efficiency, we only select sub-models with comparable pricing (approximately $5) and exclude enterprise-level models. All pricing information is based on data retrieved in January 2026 [17]–[19].

2) Comparison of Different Attack Settings: We compare our feature selection method with a typical random selection baseline. As illustrated in Fig. 6, our method achieves a rapid increase in ASR as the number of modified features m grows, reaching nearly 100% with only eight selected features. In contrast, the random selection approach increases more gradually and requires about 15 features to achieve a comparable success rate. This result shows the effectiveness of the embedded domain knowledge and the proposed update policy. Since the ASR tends to saturate when $m \geq 10$, the maximum value of m in the following is limited to 10. Fig. 7 presents the distribution of feature selection frequencies. Our method clearly identifies and prioritizes the most influential features, leveraging them to generate high-fidelity adversarial data. In contrast, the random selection method treats all features equally, resulting in a uniform distribution.

TABLE II: Comparison of different model configurations.

Configuration		API Calls	Error Rate (%)	Price (1M tokens)
Generation A_g	**Optimization A_e**			
GPT-5.1	GPT-5.1	23	26.1	$1.25
GPT-4o	**GPT-4o**	19	0.0	$2.50
GPT-5.1	GPT-4o	54	24.1	–
GPT-4o	GPT-5.1	29	3.4	–
Opus4.6	Opus4.6	43	27.9	$5.00
Sonnet4.5	Sonnet4.5	53	0.0	$3.00
Opus4.6	**Sonnet4.5**	46	0.0	–
Sonnet4.5	Opus4.6	55	0.0	–
Gemini-3-pro	**Gemini-3-pro**	67	0.0	$2.00

TABLE III: System disruption rates (%) vs. attack methods.

Defense Strategy	LEB (%)	GPT (%)	Claude (%)	Gemini (%)
LOF	65	95	95	94
EmpCov	66	81	83	81
OCSVM	80	80	83	80
Outlier	72	93	92	92
fzKNN	82	85	88	85

3) Impact of IFM: We leverage a LOF-based discriminator to evaluate the effectiveness of the IFM block. Fig. 8 compares the SDR with and without IFM under different number of features modified m. As shown, IFM consistently increases the SDR, particularly when m is small. For example, when $m = 3$, the improvement reaches about 22.73%, which can be attributed to the reduction of repeated failures due to the use of IFM. Moreover, with IFM the attack achieves a similar SDR at $m = 4$ compared to $m = 6$ without IFM, indicating that IFM effectively accelerates convergence. As m increases, the performance gap gradually narrows, suggesting that IFM is most beneficial when the modification budget is limited.

GPT

Guideline:
1) Modify only a minimal, highly selective subset of the entries, specifically constrained to the provided positions.
2) The modification must preserve plausibility and stealth while being strong enough to cause the SVM-based global decision model to misclassify.
3) Prefer modifications on features that significantly impact the global decision outcome.
4) Use the failed example as a negative benchmark:
[3.65, -2.54, 0.91, -8.83, 3.6, -10.67, -2.49, 5.95, 1.6, -14.16, -0.22, 0.19, 2.59, 2.32, 4.25, -2.97, -1.58, -2.3, -10.73, 15.0]. Avoid this approach as it compromises stealth.
5) Balance stealth and impact: modifications must induce misclassification without attracting detection.

Claude

Guideline:
1) Modify ONLY the specified position(s) with controlled, realistic perturbations that maintain feature plausibility and stealth.
2) Each modification must be sufficiently strong (magnitude of change typically between 3-8 units) to induce misclassification by the SVM-based global decision model, but not so extreme as to trigger defenses.
3) Avoid trivial changes (< 2 units) that preserve original classification, and avoid excessive changes (> 15 units) that break realism.
4) Prioritize positions with high global influence on the decision boundary while preserving local feature distribution consistency.

Defense Failure Reference:
The following example FAILED by triggering defenses due to excessive perturbation:
[3.65, -2.54, 0.91, -8.83, 3.6, -10.67, -2.49, 5.95, 1.6, -14.16, -0.22, 0.19, 2.59, 2.32, -18.5, -2.97, -1.58, -2.3, -10.73, -12.34]
(Position 15 changed from 4.25 to -18.5: magnitude 22.75 - TOO EXTREME)
Apply a moderate, defense-evading perturbation at position 15 that balances stealth with effectiveness.

Gemini

Failure Reference (Defense Triggered):
The following adversarial attempt failed because the perturbations were too drastic/obvious:
[3.65, -2.54, 0.91, -8.83, -2.85, -10.67, -2.49, 5.95, 1.6, -14.16, -0.22, 0.19, 2.59, 2.32, -3.1, -2.97, -1.58, -2.3, -10.73, -12.34]
Note: Changing index 5 (from 3.6 to -2.85) and index 15 (from 4.25 to -3.1) created statistical outliers that were detected.

Revised Guideline:
1) **Enhanced Stealth**: Modify indices 5 and 15 with perturbations smaller than the failed example. The new values must remain plausible within the context of the data distribution (e.g., simulating fading or noise rather than data corruption) to bypass outlier detection.
2) **Maintained Effectiveness**: Despite the stricter stealth constraint, the changes must still be sufficiently directional to push the global SVM decision across the boundary. Do not make the changes so trivial that the classification remains unchanged.

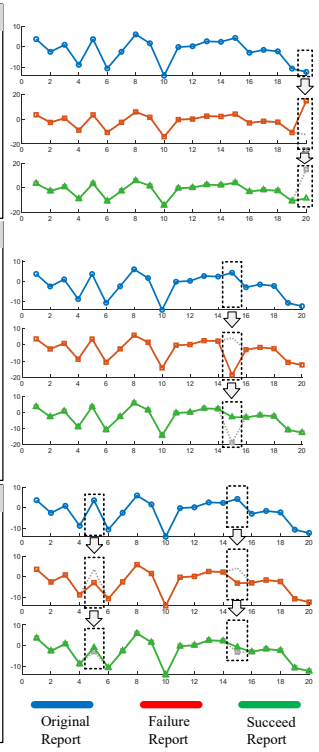

Fig. 9: Examples of prompt optimization under different models for the identical sensing report [3.65, -2.54, 0.91, -8.83, 3.6, -10.67, -2.49, 5.95, 1.6, -14.16, -0.22, 0.19, 2.59, 2.32, 4.25, -2.97, -1.58, -2.3, -10.73, -12.34].

4) Attack Impacts on Defense Strategies: Table I reports the SDR for various defense strategies. First, Table I shows that as the number of malicious nodes m increases, the SDR consistently rises across all defense strategies and LLM backends. This is because when a larger number of nodes are compromised, attackers can collaboratively manipulate the overall distribution of sensing reports. As a result, statistical anomaly detection methods have limited reliable ground-truth data to distinguish between benign and malicious reports, thereby increasing the SDR. We also observe that when $m \geq 6$ the increase in SDR becomes less significant. This is because six malicious nodes already provide sufficient capability to manipulate the overall sensing environment. Therefore, further increasing m yields only marginal additional benefits for the attacker. Among the three LLM models, Claude achieves the best attack performance, reaching an SDR of 98.35% under the LOF defense. In contrast, among the three defense strategies, OCSVM yields the lowest SDR across all LLM models.

5) Practical Examples: In Fig. 9, we provide detailed examples for each LLM model when modifying an identical original sensing report for the attack. For each model, we present the updated prompt generated by A_e, the original channel sensing report, the failed report, and the successful report to illustrate how the prompt is iteratively refined from failure to success. From Fig. 9, we observe that different LLM models generate noticeably different updated prompts during the refinement process. GPT tends to produce concise guidelines that emphasize minimal and targeted perturbations, often in a more general manner without explicitly constraining the modification scope. For example, in one guideline, "The modification must preserve plausibility and stealth while being strong enough," the instruction provides the qualitative requirements on stealth and plausibility but does not provide a quantitative specification of the allowed modification magnitude. In contrast, Claude often provides more detailed instructions with explicit constraints on the perturbation magnitude and feature plausibility. For instance, it specifies conditions such as "magnitude of change typically between 3-8 units," "avoid trivial changes ($<$2 units)," and "avoid excessive changes ($>$15 units)." These explicit constraints define a precise range for the modifications, which leads to more controlled and consistent perturbation generation. Gemini exhibits an intermediate behavior between GPT and Claude. For example, Gemini provides illustrative guidance such as "(e.g., simulating fading or noise rather than data corruption)" to further justify the modification strategy. Although this example is not quantitatively precise, it still offers more concrete guidance than GPT, which typically provides high-level instructions without specific examples.

6) Evaluating the Attacks: To evaluate the effectiveness of our attack model, we simulate a FC where all generated fake reports that successfully bypass the discriminator are forwarded for final decision-making. For comparison, we also implement the LEB attack [8], which is the most related to our setting. As shown in Table III, our attack demonstrates a clear advantage over the LEB method. In particular, our attack achieves SDRs of $94-95\%$ under LOF and $81-83\%$ under EmpCov, significantly outperforming the 65% and 66% achieved by the LEB method. To verify the generalization capability, we evaluate it against two unseen defense strategies:

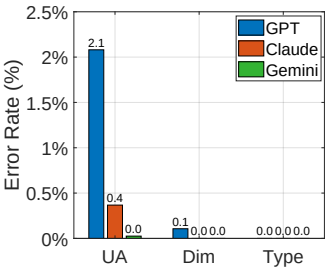

Fig. 10: Error rate versus error types.

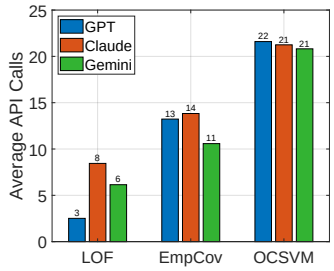

Fig. 11: Computational cost for different defenses.

TABLE IV: Per-Sample computation cost (in US dollar) under different defense strategies.

Model	LOF	EmpCov	OCSVM	Average
GPT	0.0129	0.0203	0.0914	0.0415
Claude	0.0431	0.0661	0.1275	0.0789
Gemini	0.0204	0.0397	0.0677	0.0426

a statistics-based defense, the Outlier Factor–based defense (Outlier) [31], and a machine learning–based defense, the fuzzy kNN–based defense (fzKNN) [32]. Although these defenses are not used during the training process, our attack still achieves high SDRs. Specifically, the SDR reaches approximately 92-93% under the Outlier defense and 85-88% under the fzKNN defense, both consistently higher than the corresponding results of the LEB method.

7) Evaluating the Error Detection: In Fig. 10, overall error rates are low across all LLM backends, with UA dominating the other two error types. Also, GPT shows a slightly higher UA rate ($\approx 2.1\%$) than Claude ($\approx 0.4\%$) and Gemini ($\approx 0\%$). This is because GPT-4o does not consistently enforce the specified constraints; thus, the update process may unintentionally modify other positions, leading to a large UA errors.

8) Computation Cost Analysis: For the computational overhead, we measure the average number of LLM API calls required to successfully launch an attack [33], [34] under different defense strategies. As shown in Fig. 11, under the OCSVM defense, all LLM models require significantly more queries than under the other two defenses, indicating the stronger capability of machine learning-assisted classification methods in detecting malicious sensing reports. Note that the number of API calls varies across different LLM models under the LOF and EmpCov defenses; however, they become very similar under the OCSVM defense. This is because the maximum number of prompt-updating iterations is limited by the upper bound $T_{\max}$. The similar number of API calls therefore suggests that all LLM models fail to bypass this defense within the allowed iterations and eventually reach the iteration limit. For an intuitive understanding of the cost, the corresponding monetary cost (in US dollars) for successfully attacking a single data instance is summarized in Table IV. Overall, the average cost per attack is approximately \$0.05, indicating that current LLM APIs provide a strong capability for generating malicious samples at very low cost.

V. Related Works

Attacks in Spectrum Sensing: Various attacks have been developed against spectrum sensing–based networks and applications, such as cognitive radio systems [35]–[38], where adversaries generate malicious sensing reports to manipulate the final spectrum access decision. Recent studies have also explored the use of machine learning techniques to generate adversarial or malicious sensing samples [8]. However, none of them have investigated the feasibility of directly leveraging LLMs to generate malicious samples, thereby avoiding the computationally expensive model training process.

LLM in Wireless Security: LLMs have been widely adopted in various security–related tasks from both offensive and defensive perspectives [39], [40]. For example, on the defensive side, LLM-based threat detection systems have been developed by analyzing network data across different wireless networking applications [40]–[44]. On the offensive side, LLMs have also been incorporated into various attack frameworks targeting cyber systems, such as prompt injection attacks [10], [11], membership inference attacks [12], and data poisoning attacks [13], [14]. However, limited attention has been paid to wireless networking scenarios. Authors in [45] introduce code injection attacks in IoT systems, and [46], [47] explore vulnerabilities caused by adversarial prompts in mobile agents. Nevertheless, these studies largely overlook the efficiency and adaptive design for rapidly changing wireless environments.

Prompt Engineering in Attacks: Prompt engineering in attacks enables LLMs to generate malicious content through carefully crafted adversarial prompts. For example, template-based adversarial prompts leverage predefined prompt structures to trigger undesired behaviors [48]–[50]. Chain-of-Thought (CoT) exploitation guides the model through step-by-step reasoning processes that gradually lead to restricted or sensitive outputs [51]. In addition, multi-turn prompting attacks iteratively refine prompts across multiple interactions with the model to circumvent safety mechanisms [21], [52]. However, existing studies primarily focus on general LLM vulnerabilities and have overlooked the incorporation of wireless domain knowledge into prompt design.

VI. Conclusion

In this paper, we proposed an LLM-powered multi-agent attack framework targeting the CSS system. Our method integrates an LLM into the attack pipeline to support both perturbation generation and evaluation, significantly improving the attack success rate against various existing defenses. In addition, we incorporated wireless-domain knowledge into the engineering to enhance the generation of malicious reports. We implemented the proposed attack framework in TV white space systems using LLM APIs from different providers. Experimental results show that our approach achieves SDR up to 98.35% against several defenses. Moreover, our results indicate that the average cost of generating each malicious sensing report is only about \$0.05, highlighting the urgent need to develop advanced prompt sanitization methods to prevent the abuse of the powerful capabilities of LLMs.

Acknowledgement: The work at OU was supported in part by NSF Grants 2316720, 2321271, and 2544044. The work at NCSU was supported in part by NSF Grant 2350075.

REFERENCES

[1] H. Venkataraman and G.-M. Muntean, *Cognitive radio and its application for next generation cellular and wireless networks*. Springer, 2012, vol. 5.

[2] J. Lunden, V. Koivunen, and H. V. Poor, "Spectrum exploration and exploitation for cognitive radio: Recent advances," *IEEE Signal Process. Mag.*, vol. 32, no. 3, pp. 123–140, 2015.

[3] R. Mei and Z. Wang, "Deep learning-based wideband spectrum sensing: A low computational complexity approach," *IEEE Commun. Lett.*, vol. 27, no. 10, pp. 2633–2637, Oct. 2023.

[4] A. Saeed, K. A. Harras, E. Zegura, and M. Ammar, "Local and low-cost white space detection," in *Proc. IEEE ICDCS*, 2017.

[5] H. Chen, M. Zhou, L. Xie, and J. Li, "Cooperative spectrum sensing with m-ary quantized data in cognitive radio networks under ssdf attacks," *IEEE Trans. Wireless Commun.*, vol. 16, no. 8, pp. 5244–5257, 2017.

[6] C. Szegedy, "Intriguing properties of neural networks," *arXiv*, 2013.

[7] T. Hou, S. Bi, T. Wang, Z. Lu, Y. Liu, S. Misra, and Y. Sagduyu, "Muster: Subverting user selection in mu-mimo networks," in *Proc. IEEE INFOCOM*, 2022.

[8] Z. Luo, S. Zhao, Z. Lu, J. Xu, and Y. E. Sagduyu, "When attackers meet ai: Learning-empowered attacks in cooperative spectrum sensing," *IEEE Trans. Mobile Comput.*, vol. 21, no. 5, pp. 1892–1908, May 2022.

[9] L. Schwinn, "Adversarial attacks and defenses in large language models: Old and new threats," *arXiv*, 2023.

[10] Y. Liu, Y. Jia, R. Geng, J. Jia, and N. Z. Gong, "Formalizing and benchmarking prompt injection attacks and defenses," in *Proc. USENIX Security*, 2024.

[11] S. Rossi, A. M. Michel, R. R. Mukkamala, and J. B. Thatcher, "An early categorization of prompt injection attacks on large language models," *arXiv*, 2024.

[12] W. Fu, H. Wang, C. Gao, G. Liu, Y. Li, and T. Jiang, "Membership inference attacks against fine-tuned large language models via self-prompt calibration," in *Proc. NeurIPS*, 2024.

[13] P. He, H. Xu, Y. Xing, H. Liu, M. Yamada, and J. Tang, "Data poisoning for in-context learning," in *Proc. NAACL*, 2025.

[14] A. Das, A. Tariq, F. Batalini, B. Dhara, and I. Banerjee, "Exposing vulnerabilities in clinical llms through data poisoning attacks: Case study in breast cancer," *MedRxiv*, 2024.

[15] A. Sherstinsky, "Fundamentals of recurrent neural network (rnn) and long short-term memory (lstm) network," *Physica d: Nonlinear phenomena*, vol. 404, p. 132306, 2020.

[16] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz *et al.*, "Transformers: State-of-the-art natural language processing," in *Proc. EMNLP*, 2020.

[17] OpenAI, "Openai api usage pricing," available: https://platform.openai.com/docs/pricing.

[18] Anthropic, "Claude api pricing documentation," available: https://docs.anthropic.com/en/docs/about-claude/pricing.

[19] Google DeepMind, "Gemini api pricing guide," available: https://ai.google.dev/pricing.

[20] L. Gao, A. Madaan, S. Zhou, U. Alon, P. Liu, Y. Yang, J. Callan, and G. Neubig, "Pal: Program-aided language models," in *Proc. ICML*, 2023.

[21] N. Shinn, F. Cassano, E. Berman, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language agents with verbal reinforcement learning," *arXiv*, 2023.

[22] C. Yan, J. Wang, L. Zhang, R. Zhao, X. Wu, K. Xiong, Q. Liu, G. Kang, and Y. Kang, "Efficient and accurate prompt optimization: the benefit of memory in exemplar-guided reflection," in *Proc. ACL*, 2025.

[23] M. Renze and E. Guven, "Self-reflection in llm agents: Effects on problem-solving performance," *arXiv*, 2024.

[24] W. Zhong, L. Guo, Q. Gao, H. Ye, and Y. Wang, "Memorybank: Enhancing large language models with long-term memory," in *Proc. AAAI*, 2024.

[25] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv*, 2015.

[26] Federal Communications Commission (FCC), "Amendment of part 15 of the commission's rules for unlicensed operations in the television bands," Tech. Rep., 2014.

[27] F. Amini and M. Mahdavi, "Local outlier factor based cooperation spectrum sensing scheme for defending against attacker," in *Proc. IST*, 2014.

[28] Y. Zeng and Y.-C. Liang, "Spectrum-sensing algorithms for cognitive radio based on statistical covariances," *IEEE Trans. Veh. Technol.*, vol. 58, no. 4, pp. 1804–1815, May 2009.

[29] S. Rajasegarar, C. Leckie, and M. Palaniswami, "Pattern based anomalous user detection in cognitive radio networks," in *Proc. IEEE ICASSP*, 2015.

[30] I. M. Suliman, J. Lehtomäki, and K. Umebayashi, "On the effect of false alarm rate on the performance of cognitive radio networks," *J. Wireless Com. Netw.*, 2015.

[31] P. Kaligineedi, M. Khabbazian, and V. K. Bhargava, "Malicious user detection in a cognitive radio cooperative sensing system," *IEEE Trans. Wireless Commun.*, vol. 9, no. 8, pp. 2488–2497, Aug. 2010.

[32] H. Wang and Y.-D. Yao, "Primary user boundary detection in cognitive radio networks: Estimated secondary user locations and impact of malicious secondary users," *IEEE TVT*, vol. 67, pp. 4577–4588, 2018.

[33] C. Guo, J. Gardner, Y. You, A. G. Wilson, and K. Weinberger, "Simple black-box adversarial attacks," in *Proc. ICML*, 2019.

[34] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. R. Narasimhan, and Y. Cao, "React: Synergizing reasoning and acting in language models," in *Proc ICLR*, 2022.

[35] A. S. Rawat, P. Anand, H. Chen, and P. K. Varshney, "Collaborative spectrum sensing in the presence of byzantine attacks in cognitive radio networks," *IEEE TSP*, vol. 59, no. 2, pp. 774–786, 2010.

[36] M. Liu, H. Zhang, Z. Liu, and N. Zhao, "Attacking spectrum sensing with adversarial deep learning in cognitive radio-enabled internet of things," *IEEE Transactions on Reliability*, vol. 72, pp. 431–444, 2022.

[37] Y. Li and Q. Peng, "Achieving secure spectrum sensing in presence of malicious attacks utilizing unsupervised machine learning," in *Proc. IEEE MILCOM*, 2016.

[38] J. Wu, P. Li, Y. Chen, J. Tang, C. Wei, L. Xia, and T. Song, "Analysis of byzantine attack strategy for cooperative spectrum sensing," *IEEE Communications Letters*, vol. 24, no. 8, pp. 1631–1635, 2020.

[39] 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.

[40] B. Singer, K. Lucas, L. Adiga, M. Jain, L. Bauer, and V. Sekar, "On the feasibility of using llms to execute multistage network attacks," *arXiv*, 2025.

[41] H. Wen, P. Sharma, V. Yegneswaran, P. Porras, A. Gehani, and Z. Lin, "6g-xsec: Explainable edge security for emerging openran architectures," in *Proc. ACM HotNets*, 2024.

[42] H. Zhang, A. B. Sediq, A. Afana, and M. Erol-Kantarci, "Large language models in wireless application design: In-context learning-enhanced automatic network intrusion detection," in *Proc. IEEE GLOBECOM*, 2024.

[43] F. Adjewa, M. Esseghir, and L. Merghem-Boulahia, "Efficient federated intrusion detection in 5g ecosystem using optimized bert-based model," in *Proc. IEEE WiMob*, 2024.

[44] K. B. Kan, H. Mun, G. Cao, and Y. Lee, "Mobile-llama: Instruction fine-tuning open-source llm for network analysis in 5g networks," *IEEE Network*, vol. 38, no. 5, pp. 76–83, 2024.

[45] H. A. Noman and O. M. Abu-Sharkh, "Code injection attacks in wireless-based internet of things (iot): A comprehensive review and practical implementations," *Sensors*, vol. 23, no. 13, p. 6067, 2023.

[46] C. Du, Q. Huang, T. Tang, Z. Wang, A. Nadkarni, and Y. Xiao, "Measuring the security of mobile llm agents under adversarial prompts from untrusted third-party channels," *arXiv*, 2025.

[47] L. Wu, C. Wang, T. Liu, Y. Zhao, and H. Wang, "From assistants to adversaries: Exploring the security risks of mobile llm agents," *arXiv*, 2025.

[48] A. Pei, Z. Yang, S. Zhu, R. Cheng, and J. Jia, "Selfprompt: Autonomously evaluating llm robustness via domain-constrained knowledge guidelines and refined adversarial prompts," in *Proc. COLING*, 2025.

[49] B. Wang, Y. Deng, R. Luo, P. Liang, and T. Bi, "Mplinker: Multi-template prompt-tuning with adversarial training for issue–commit link recovery," *Journal of Systems and Software*, vol. 223, p. 112351, 2025.

[50] X. Deng, J. Wu, M. Chen, Y. Xiao, K. Xu, and Q. Li, "Automating agent hijacking via structural template injection," *arXiv*, 2026.

[51] Y. Wang, Y. Yu, J. Liang, and R. He, "A comprehensive survey on trustworthiness in reasoning with large language models," *arXiv*, 2025.

[52] A. Reddy, A. Zagula, and N. Saban, "Autoadv: Automated adversarial prompting for multi-turn jailbreaking of large language models," *arXiv*, 2025.