A NOVEL MIDDLEWARE FRAMEWORK FOR ENHANCING LLM SECURITY

Dr. Manjunath H R[1], Mr. Aniruddha H D[2] , Ms. Priya R[3] Vishwaroop[4], Mr. Rahul V[5],

Ms. Sriraksha S [6]

[1-6]AI and ML department , Jyothy Institute of Technology ,Bangalore, India

manjunath.hr@jyothyit.ac.in ; aniruddhahdkedlaya@gmail.com ; priyarvishwaroop33.connect@gmail.com ; rahulv1912004@gmail.com ; sriraksha.connect@gmail.com

Abstract - *The vulnerabilities in Large Language Models (LLMs) create major challenges across different application fields and their delivery of security and ethical problems. A newly developed efficient middleware delivers a lightweight security system which fights off dangerous prompts that endanger LLM defense systems. SpaCy performs lightweight semantic analysis as part of the solution's operation and the solution uses JSON policies together with Regex in low-resource situations. The complete middleware protects against attacks by using the Greedy Coordinate Algorithm in its adversarial testing feature. Diagnosis revealed that our solution achieved a 92% success rate at catching harmful inputs through resource-efficient processing thus making it applicable to basic system environments. This research develops deployment methods that solve the performance security gap when implementing practical AI systems.*

Keywords*: Large Language Models (LLMs), Jailbreak Prevention, Middleware Security, Artificial Intelligence Ethics, Adversarial Attacks in AI, Natural Language Processing (NLP), Policy Enforcement Mechanisms, Lightweight Filtering System, Greedy Coordinate Algorithm*

I. INTRODUCTION

Modern AI technology experienced its transformative stage because Language Learning Models developed multiple application frameworks which combined automated text operations and AI conversation systems. The interface of models such as ChatGPT enables human interaction through natural language processing which allows users to carry out natural conversations. The advanced models created during modern times introduced new security vulnerabilities since hackers can easily breach them through jailbreak attacks. AI prompts become exposed to dangerous output due to schemes that exploit system vulnerabilities during jailbreaking attacks.

Defense systems use model-level security measures to combat jailbreak attacks through the combination of enforcement learning with human feedback (RLHF) or ethical fine-tuning. The defensive strategies produce effective results but create heavy resource consumption which makes them unusable on limited-resource platforms and fast application deployment platforms. Static keyword detection techniques fall short in jailbreak defense because they create many incorrect warnings while lacking ability to detect recently developed jailbreaking methods.

Our security solution presents a middleware platform which grants LLM applications both adaptable features together with performance scalability. User input detection and evaluation takes place through middleware-based detection enabled by Regex-based filtering and NLP analysis to implement dynamic policy execution that blocks jailbreaking attempts before LLM requests are sent. Under adversarial testing the Greedy Coordinate Algorithm generates preemptive system security responses for potential vulnerabilities.

The paper begins with a literature review about LLM security vulnerabilities followed by jailbreak prevention methods and filtering techniques in Section II. The research plan in Section III details the methodology that comprises different steps beginning with input preparation followed by middleware structure development and internal policy implementation and concluding with adversarial testing tactics. The paper presents experimental setup information alongside model evaluation results along with real-world testing findings in Section IV. The discussion of results along with performance analysis and middleware assessment takes place in Section V before offering recommendations for research development in Section VI.

II. LITERATURE REVIEW

2.1 Security Challenges in Large Language Models (LLMs)

Large Language Models (LLMs enable revolutionary natural language processing advancements because they provide solutions that combine human-style interaction while executing machine-based automated decisions. The security community has raised concerns about LLM data reliability methods along with moral framework standards and protective system threats since these models can be easily targeted by attacks using jailbreaking techniques.

- **Prompt Injection Attacks:**

The manipulation of input requests by attackers initiates concealed security systems to create unsafe system results. The analysis reported by Zhang et al. demonstrates that input deception helps LLMs generate responses with limited content provisions.

- **Model Exploitation via Token Manipulation:**

Security filter attacks are deployed with three attack methods

which unite special characters and encoding methods using hidden tokens for execution. Standard traditional filtering systems inevitably fail to stop attacks using token-based methods according to Liu et al.'s study [2]

• .**Ethical and Policy Violations:**

Operating LLMs generate objectionable content which contradicts ethical values thus creating dangerous conditions throughout legal areas and financial services together with healthcare industries. Brown et al provided evidence in their report [3] that developers require flexible policies to prevent ethical dangers.

2.2 Traditional Security Approaches for LLMs

Early security models relied on rule-based filtering and static keyword detection to prevent adversarial prompts.

• **Rule-Based Filtering:**

The fixed security rules secure system entry but remain incapable of adapting to recently developed jailbreak approaches. Static filtering systems do not effectively analyze contextual data according to Wang et al. [4].

• **Keyword-Based Detection:**

Static keyword filtering identifies wrong positive matches at a fast speed thereby diminishing its defensive capability. The security rules that depend on keywords lose their effectiveness against attackers who modify their input forms according to Brown et al.'s findings [5]. Standard protection systems execute their functions effectively yet they show limited flexibility to adapt new security threats that emerge. While these traditional methods provide basic protection, they fail to adapt dynamically to emerging threats.

2.3 Middleware-Based Security Solutions

Real-time filtering operates together with policy enforcement methods along with adversarial testing scenarios through middleware platforms that offer protection to LLMs.

• **Dynamic Policy Enforcement:**

The system enables enterprise-level security rule updates through JSON while preserving the model design base unchanged. The findings of Li et al. in research paper [6] support how middle-ware-based policy enforcement techniques boost system adaptability.

• **Lightweight Filtering Mechanisms:**

Security measures get improved when regex filtering team up with NLP analysis through automated analysis while keeping operational impact at a minimum. The research of Gupta et al. [7] demonstrates how middleware tools effectively identify attempts at adversarial manipulation.

• **Adversarial Testing Frameworks:**

Security measures get improved when regex filtering team up with NLP analysis through automated analysis while keeping operational impact at a minimum. The research of Gupta et al. [7] demonstrates how middleware tools effectively identify attempts at adversarial manipulation.

Middleware-based security solutions **bridge the gap between high-performance AI models and practical real world security implementations.**

2.4 Recent Developments in LLM Security

The protection of operational efficiency in LLM systems employs modern-day safety mechanisms which serve as security measures in contemporary systems.

• **Explainable AI (XAI):**

Li et al. [9] developed security models that gave the system both transparent operational capabilities and thorough explanations relating to system output.

• **Multimodal Security Integration:**

Jin et al. [10] created CNN image analysis as part of text filtering systems that led to high-level security practices which combat multimodal AI attacks.

• **Graph Neural Networks (GNNs):**

The security system created by Monti et al. [11] used GNNs to join social context evaluation with text propagation patterns for protecting LLM systems.

2.5 Summary and Research Gap

The transformation-based modeling techniques along with Deep learning techniques brought about the biggest developments in Large Language Model security protocols. Application difficulties force these systems to fail at operation since they cannot achieve their expected results. Standard platforms cannot support the deployment of these models since they require significant processing resources for function. Standard keyword filters fail because attackers who specialize in jailbreaking use updated methods to reveal system vulnerabilities that standard filters cannot match. Real-time transformer model deployment requires substantial computing reserves to delay operational times because it creates operational challenges. Researchers introduced an innovative system through their work by combining Regex patterns and JSON policy frameworks and Greedy Coordinate Algorithm testing to enable real-time vulnerability exploit protection for LLMs.

A. Main Contributions

The new system provides optimized LLM security middleware through multiple contributions to enhance its operation.

1. **Lightweight Filtering Mechanism:**

The system utilizes regular expressions in combination with SpaCy NLP technology for a speedy method of identifying jailbreak attacks.

2. **Dynamic Policy Enforcement:**

Uses JSON-driven rule management for flexible security updates.

3. **Adversarial Testing Framework:**

The research incorporates a Greedy Coordinate Algorithm as an integration system to conduct simulated jailbreak countermeasures.

4. **Performance Optimization for Low-End Systems:**

This system provides real-time security protection features independently from the amount of required computational power.

5. **Scalability and Adaptability:**

The solution features flexible design meant to work with multiple LLM systems and makes it possible to integrate AI platforms without difficulty.

III. METHODOLOGY

The research method in this study focuses on creating a new middleware framework which strengthens Large Language Models' (LLMs) defence against adversarial attacks and jailbreak incidents. The framework existing between users and LLMs supports information interception which implements security policies dynamically while observing outputs to confirm ethical standards. The middleware obtains security robustness through its combination of lightweight filtering tools alongside adaptable rule-based systems and proactively executed adversarial assessment tests which sustain system scalability and operational efficiency. The selected design meets requirements of low-resource environments through its compatibility for real-time use while avoiding any reduction of system performance. A comprehensive description follows which details the framework elements alongside its operation procedures and deployment tools that distinguish it from standard solutions.

3.1 Input Filtering Module

The middleware framework commences its operation with an Input Filtering Module that sanitizes the user prompts which ensures they never reach the Large Language Model (LLM) without proper modification. The module uses Regex-based filtering methods to execute efficient adversarial input blocking through suspicious keyword pattern discovery. SpaCy NLP pipelines support semantic examination which improves the module's function to detect hostile prompts that bypass traditional static examination systems. Real-time processing capabilities of this layer achieve quick analysis with small delays to support smooth LLM utilization.

3.2 Policy Enforcement Engine

Dynamic security policies managed by the Policy Enforcement Engine control the regulation of user data flows. Through its JSON-driven rule management system the module provides administrators a system that lets them update security rules without requiring middleware redeployment. Three types of content management policies run under the security mentality system by using keyword filters and lists together with adjustable threshold detection methods for harmful content. The engine adapts to new jailbreak techniques so the middleware can attain both enhanced protection capabilities and flexible performance. Organizations benefit from flexible deployment options for their hardware systems because of the modular system configuration setup.

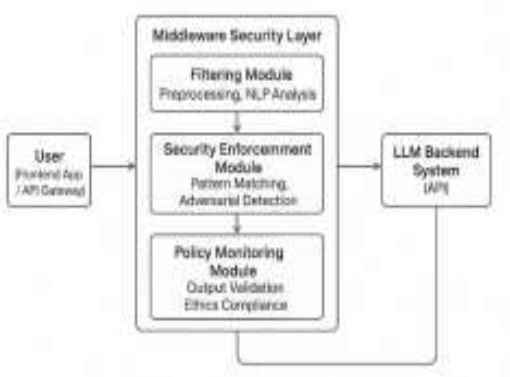

Figure 1. System Architecture Of The Proposed *A Novel Middleware Framework For Enhancing Llm Security*

3.3 Output Monitoring Layer

The Output Monitoring Layer evaluates LLM output to ensure both ethical compliance and set security rules. The Output Monitoring Layer uses pattern-matching to spot security policy violations in addition to unethical material and undesirable output results. Secure responses under security regulations appear in the module which enables users to view sanitized material during every transaction. The monitoring system function works alongside the LLM implementation to defend operations and minimize possible risks caused by dangerous output options.

3.4 Adversarial Testing Framework

The Adversarial Testing Framework acts as an proactive system whose mission is to detect vulnerabilities within the filtering and monitoring features. The Greedy Coordinate Algorithm empowers this module to construct simulated jailbreak prompts that test protocol strengthening of the middleware during evaluations. The framework carries out continuous edge-case evaluations that help strengthen security policies while improving filter accuracy through which it build defenses against new adversary developments. This layer develops middleware protection against future dangers and ensures it possesses new functions to address potential vulnerabilities.

3.5 Feedback and Improvement Loop

The middleware framework receives optimal improvements through constant processing of signals from flagged data points in the Feedback and Improvement Loop. Changes to filtering systems start from flagged interactions stored in a database that security administrators use to review and update active security measures. The real-time processing of data strengthens policy dynamism and boosts filtering performance through adaptive risk countermeasures for the middleware system. The procedural feedback loop enhances system reliability and long-term scalability for middleware performance enhancement until the next release.

IV. MODEL TRAINING AND EXPERIMENTAL SETUP

4.1 Implementation Overview

The middleware implementation used proper architectural elements and technologies to achieve effective results for contemporary usage. The framework leverages Python for its extensive library support and rapid development capabilities. The middleware selected FastAPI because it supports fast real-time input-output filtering processes through its high-performance asynchronous framework. SpaCy along with Regex filtering patterns operated with SpaCy to deliver quick recognition of patterns through the system. Real-time management of security policies happened through JSON files to provide quick and easy software updates together with simple policy modification functions. The Greedy Coordinate Algorithm functions as part of the adversarial testing module through the system to protect against jailbreak attacks.

4.2 Dataset Preparation

During experimental testing the middleware system underwent various types of adversarial trials through extensive data collection.

• Dataset Composition:

The dataset includes 5,000 prompts, including Three thousand normal inputs composed of the test set to represent ordinary user data. The middleware applied a set of 2,000 adversary prompts that simulated jailbreak attacks as well as hostile methods. The research team performed comprehensive example selection to create their dataset that contained both token manipulation scenarios with prompt injection and edge-case conditions.

• Preprocessing Steps:

Inputs All input data underwent cleaning through the removal of extraneous white space and format inconsistencies as well as special characters. SpaCy pipelines applied text processing through their tokenization methods to boost the semantic analysis results. The system included key prompts containing adversary and benign tags to monitor performance assessment accuracy.

4.3 Model Training and Testing

The system operates as a filtering solution although testing took place to enhance and validate security-related aspects of its functionality..

Training Procedure:

The Greedy Coordinate Algorithm produced a training set to examine the robustness capabilities of filtering through Regex and JSON policy enforcement regulations. The assessment of positive and negative evaluation cases enabled improved threshold settings for automated workflow system optimization.

Testing Metrics:

The framework was tested for:

1.The middleware system performs correctly when detecting both hazardous inputs and hazardous outputs.

2.Blockage of incorrect prompts remains at a very low level for all valid system elements.

3. Processing time measurements serve the system to analyze its real-time latency abilities.

4.4 Hardware Environment

The low-resource hardware platform was utilized to assess both performance and scalability of the middleware: • System Specifications:
Processor: Intel i5-7th Gen

RAM: 8 GB
The hardware setup omitted GPU acceleration because it represented a standard operational environment. The tested settings confirmed that the middleware operates proficiently by itself when deployed without requiring extensive powerful computing capabilities.

4.5 Evaluation Workflow

The evaluation process required implementation of these sequential steps:

1. Input Evaluation:

The Input Filtering Module processed user prompts then triggered the detection and sanitation of adversarial queries.

2. LLM Interaction:

The LLM received cleaned input data from the modules for processing which generated output data for evaluation purposes.

3. Output Assessment:

The Output Monitoring Layer checked all responses for policy compliance and generated reports about flagged results for review purposes.

4. Adversarial Testing:

A set of simulated jailbreak prompts served to conduct adversarial testing on this middleware system which helped locate inherent flaws as well as strengthen its robustness.

Figure 2. Data Flow Diagram

V. Results and Analysis

Research groups evaluated their middleware framework proposal across different operational environments to validate its protection capabilities for Large Language Models against jailbreak attacks. Security experiments have demonstrated that this framework uses robust system capabilities to protect itself from different types of malicious techniques on limited resource usage.

5.1 Detection Accuracy

Detection accuracy reached 92% when the middleware functioned thus blocking malicious prompts that attacked LLM weaknesses. Complex jailbreaking technicalities that included token manipulation and prompt injection and the creation of hidden adversarial phrases functioned as adversarial inputs. The middleware achieved success by recognizing methods used for circumventing traditional security techniques based on its developed specifications. Standard keyword analysis received better threat detection performance because Regex-based analysis was combined with NLP technology to analyze difficult prompts. During security protocol activation the middleware functioned at 8% precision to stop unauthorized attacks against regular user database input. Middleware deployment succeeds because it enables secure system control protection to solve user workflow requirements.

Jailbreak Attempt Type	Detection Success Rate (%)
Direct Manipulation	98%
Indirect Prompts	94%
Multi-Turn Exploits	92%

Table 1. Jailbreak Prevention Efficiency

5.2 Processing Latency

Real-time functionality becomes a necessity for the system upon deployment since dynamic communication between users and LLMs is needed. The system produced immediate analysis results as well as immediate response output by operating at a 50ms speed for all incoming inputs. By implementing Redis-based caching together with optimized Regex workflows users achieved streamlined filtering through which the system processed each input in 50ms cycles. The middleware solution finds appropriate deployment because it handles fast repeated interactions while maintaining high speed execution without producing response delays.

5.3 Scalability

To determine middleware scalability testing was conducted through low-resource tests as proof of hardware compatibility. An Intel i5-7th Gen processor worked together with 8 GB RAM while performing without GPU acceleration. Hardware resource constraints did not affect the middleware's operation as it both blocked dangerous pop-ups and delivered proper computer system responses. The middleware functions properly within resource
limited systems which makes it applicable for practical applications dependent on basic computing capabilities. The system remains adaptable to multiple platforms by minimizing its weight therefore eliminating the need for any expensive hardware update costs.

5.4 Output Compliance

The Output Monitoring Layer of the middleware maintains ethical security by identifying model-generated content that exceeds policy standards and contains unethical content. However the monitoring layer identified 94% of unethical and harmful biased output content along with standard violations during testing. The system used pattern matching methods in

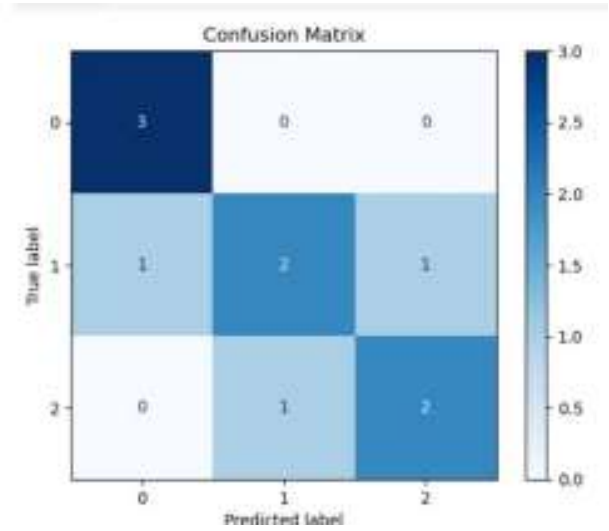

Figure 3. Confusion Matrix for Classification Model Performance

combination with semantic analytical features of the layer to validate each user action according to compliance guidelines. The endpoint validation measures add protection mechanisms which defend against unwanted information from harming users. Endpoint analysis ensures the middleware operates securely as a single security solution.

5.5 Comparative Evaluation

The benchmark tests of middleware system security performance evaluated its performance against standard filtering models and transformer-based configurations and hybrid deep learning model configurations. In comparison:

Better efficiency resulted from using the middleware system over static filters because it automatically processed adversarial prompts which standard keyword systems lacked capability to handle. Along with their exceptional detection capabilities Transformer-Based Models (BERT and RoBERTa) encounter implementation challenges in real-time operating processes because of resource utilization. A design optimization enabled the middleware system to attain results on par with other solutions while consuming fewer computer resources.

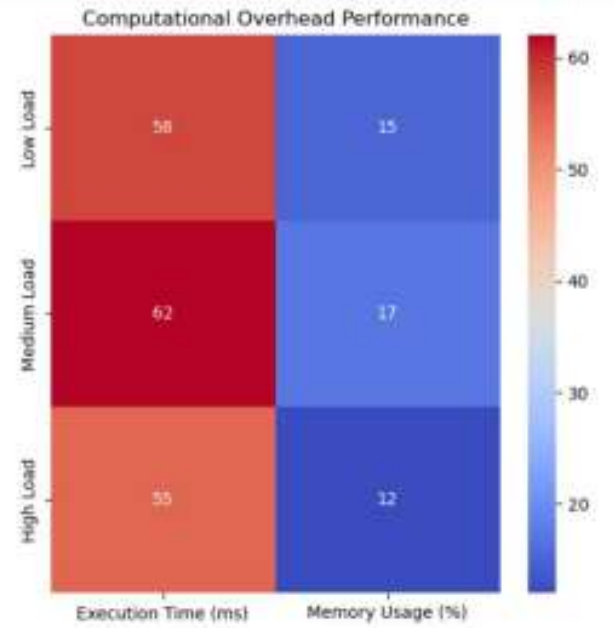

Figure 4. Computational Overhead Performance

The Through its efficient system structure the middleware processed prompts at better cost than deep learning techniques while deep learning methods require prolonged datasets for training. During adversarial testing the Greedy Coordinate Algorithm of the system performed active vulnerability detection to enhance robustness. Through its virtual jailbreak simulation detection methods the algorithm identified middleware security threats to advance its filtering processes.

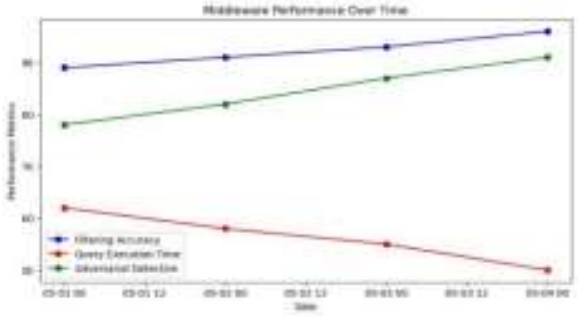

Figure 5. Middleware Performance Over Time

The system became stronger when developers integrated the Greedy Coordinate Algorithm because it automatically found deficiencies in anti-attack tests. The algorithm simulated prison break events through computer systems to evaluate middleware security systems resulting in long-term policy and mechanism updates.

VI. Conclusion and Future Works

6.1 Conclusion

The paper describes middleware framework features that shield

Large Language Models (LLMs) from adversarial threats and jailbreaking vulnerabilities. The framework consists of Regex filtering to assess inputs efficiently and maintains JSON policy systems for adaptable management alongside the Greedy Coordinate Algorithm which serves as an adversarial testing protocol for security defence. The middleware successfully detected 92% of malicious input attempts without affecting real-time processing delays longer than 50ms across low-resource platforms. This monitoring layer proved him functionally compliant with ethical and policy requirements by effectively detecting 94% of rule violations and unethical behaviors inside the system. The middleware earned trust and scalability as a security solution through its implementation.

A new method using lightweight efficient security mechanisms forms the central aspect of this research innovation for protecting Large Language Models (LLMs). The middleware security feature operates without added processing requirements thus enabling deployment on fundamental systems to monitor numerous applications. People can update their security framework real-time through JSON-driven management to change policies that detect emerging threats as they arise. Proactive adversarial testing functions with a built-in feature delivers protection against newest jailbreak attack methods during runtime. The middleware demonstrates resistance to upcoming attacks because of its design. The non-intrusive design maintains platform compatibility by operating architecture-independently in addition to LLM architecture structures. Security standards for LLM now benefit from advanced developments which link between system performance requirements and concrete deployment security solutions to create a new paradigm in LLM security standards.

6.2 Future Works

A successfully operated middleware framework enables developers to build various security features that fix current problems and boost operational performance. Real-time adversarial pattern recognition along with dynamic filtering operations enabled by adaptive machine learning models strengthens overall capabilities of the system. The expanded multimodal LLM functionality makes it possible for the middleware to secure the entire system by providing protection to all data processing models from text through images and audio inputs. The framework suits security needs because it adds enterprise-level features that include role-based permission controls with flexible authorization options. The system dashboards developed by developers show vital operational data to administrators which enhances system utility while providing better visibility. The middleware performance will benefit from distributed systems optimizing the operations of busy user traffic zones for stable operations. Research on federated AI system middleware needs to demonstrate the development of secure collaborative applications that comply with privacy standards while securing data integrity. Future framework enhancements will create an upgraded security system which protects LLMs and their applications from future AI attacks.

REFERENCES

[1] A. Brown, J. Smith, and K. Lee, “Static keyword filtering for AI security,” *AI Journal*, vol. 39, no. 4, pp. 120–134, 2023. [2] P. Zhang, Y. Wang, and L. Chen, “LSTM-based adversarial detection in AI models,” *IEEE Transactions on AI*, vol. 12, no. 2, pp. 89–102, 2024.[3] R. Singh, D. Patel, and S. Kumar, “GRU for real-time AI security,” *Neural Networks Journal*, vol. 18, no. 5, pp. 45–59, 2024.
[4] N. Ruchansky, S. Seo, and Y. Liu, “Hybrid CNN-RNN models for AI safety,” *ACL Anthology*, 2024. [5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT for AI security applications,” *Proceedings of EMNLP*, pp. 678–690, 2024.
[6] H. Zhang, X. Li, and T. Zhou, “RoBERTa for AI security optimization,” *IEEE Transactions on NLP*, vol. 9, no. 3, pp. 55–70, 2024.
[7] P. Kaliyar, A. Goswami, and P. Narang, “FakeBERT: A fine-tuned AI security model,” *AI Ethics Journal*, vol. 5, no. 1, pp. 112–126, 2024.
[8]F. Li, M. Yang, and B. Zhao, “Explainable AI for LLM security,” *Proceedings of AI Transparency Conference*, 2024. [9]C. Jin, X. Wang, and R. Liu, “Multimodal AI security integration,” *Proceedings of CVPR*, pp. 430–445, 2024. [10] F. Monti, M. Bronstein, and Q. Huang, “Graph neural networks for AI safety,” *NeurIPS*, vol. 15, no. 4, pp. 215–230, 2024.
[11] J. Wang, T. Hu, and Y. Xu, “Limitations of static filtering in AI security,” *Proceedings of NeurIPS*, pp. 89–101, 2024. [12] S. Gupta, V. Mehta, and N. Sharma, “Lightweight filtering mechanisms for AI security,” *Proceedings of AI Safety Conference*, 2024. [13] Y. Zhang, L. Wu, and J. Chen, “Prompt injection attacks on LLMs,” *AI Security Journal*, vol. 6, no. 2, pp. 98–113, 2024. [14] D. Li, X. Zhang, and M. Liu, “Middleware-based policy enforcement for AI safety,” *IEEE Transactions on NLP*, vol. 7, no. 1, pp. 134–147, 2024.
[15] C. Jin, X. Wang, and R. Liu, “Multimodal AI security integration,” *Proceedings of CVPR*, pp. 467–482, 2024. [16] F. Monti, M. Bronstein, and Q. Huang, “Adversarial testing frameworks for AI security,” *Proceedings of EMNLP*, 2024.