Prompt Injection in Large Language Model Exploitation: A Security Perspective

Jefferson Kanjirakkattu Joseph
Division Of Computer Science And Engineering
Karunya Institute of Technology and Sciences,Coimbatore, India
jeffersonjoseph2002@gmail.com

Esther Daniel
Division Of Computer Science And Engineering
Karunya Institute of Technology and Sciences,Coimbatore, India
estherdaniel@karunya.edu

V Kathiresan
Specialist – Technical,
HCL Technologies Limited, Chennai 600119, India
xyzkathir@gmail.com

Manimegalai M.A.P
Division Of f Eletronics and Communication Engineering
Karunya Institute of Technology and Sciences,Coimbatore, India
manimekalai@karunya.edu

2025 International Conference on Electronics, Computing, Communication and Control Technology (ICECCC) | 979-8-3315-2162-2/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICECCC65144.2025.11064209

***Abstract*— With the rapid growth of AI technologies, ensuring the security of open-source Large Language Models (LLMs) is crucial to maintaining their reliability and trustworthiness. The paper presents a detailed framework to evaluate the security risks of these models in today's fast-changing technological world. This framework includes generators, probes, detectors, and evaluation methods, all centered around the Prompt Inject framework, which helps identify vulnerabilities and possible attacks on LLMs. By using this approach, organizations, researchers, and developers can assess how easily these models can be manipulated and take steps to improve their security. The paper highlights important applications like security testing, penetration testing, compliance checks, and continuous monitoring, ensuring that LLMs remain safe and reliable. This research is valuable for strengthening cybersecurity efforts in the era of open-source AI, helping to build safer and more trustworthy AI systems.**

Keywords*— *Prompt Injection, Large Language Models, Security Vulnerability, Adversarial Attacks, Open-source AI, Ethical AI, Threat Detection, Prompt Security

I. INTRODUCTION

Language models are now integral to numerous technologies from chatbots to virtual assistants and text generation. However, keeping them secure is a big challenge, especially for freely available and widely used Open-Source Large Language Models (LLMs). The paper "Security Analysis of Open Source LLMs" studies the risks and challenges these models face. While LLMs have improved natural language processing, they can also be misused by hackers or bad actors. If these models are not secure, they might be tricked into revealing sensitive information, spreading harmful content, or being used for cyberattacks. Ensuring their safety is important to protect user privacy and maintain trust in AI technologies.

The suggested methodology was strategically chosen to balance theoretical robustness with practical applicability in evaluating LLM security. Transformer-based models such as GPT, BERT, and RoBERTa were selected due to their superior contextual understanding and strong track record in natural language processing tasks, especially in classification and generation. These models offer state-of-the-art performance and can be fine-tuned efficiently on domain-specific data, making them ideal for security-focused applications.

The use of curated datasets—consisting of code, prompts, and annotated vulnerabilities—ensures that our models are exposed to realistic and diverse attack scenarios. Alternative approaches like rule-based or symbolic AI techniques, while useful in static analysis, lack adaptability and contextual comprehension, limiting their effectiveness in dynamic threat landscapes.

We further incorporated custom-designed probes and pseudocode-based scanners instead of off-the-shelf vulnerability tools. This decision was driven by the need to simulate LLM-specific threats, such as prompt injection and indirect instruction manipulation, which general-purpose security tools are ill-equipped to detect. By combining dynamic probing with empirical evaluation (using metrics like F1-score and recall), our framework provides actionable insights, grounded in both quantitative rigor and real-world relevance.

The main goal of the paper is to analyze the security of Open-Source LLMs by using different tools and methods. It looks at common risks like prompt injections, where attackers manipulate a model's responses, cross-site scripting vulnerabilities, which could lead to cyberattacks, and toxic content generation, where models might produce harmful or offensive text. By testing and studying these issues, the paper aims to find weak points in LLMs and suggest ways to fix them. This research is crucial for making AI safer and preventing models from being misused.

To test security, the paper uses advanced probes like Garak, which scan LLMs for weaknesses and check how strong they are against attacks. It also includes custom tools designed to find specific security threats. By combining these tools with existing security methods, the research provides practical solutions to improve LLM security. These findings help developers, researchers, and organizations strengthen AI models, making them safer and more reliable for everyday use.

II. RELATED WORKS

In exploring the intersection between machine learning and code review processes, this research delves into various studies that illuminate how these technologies can enhance both efficiency and effectiveness. One notable study by Li et al. proposed a deep-learning model designed to classify change descriptions and code snippets, aiming to predict the acceptance of code review triplets. This model's performance highlighted the potential for deep learning to streamline the code review process. Similarly, the work of Shi et al. involved the use of convolutional and LSTM layers within their models to forecast the approval of changes in source code. Their findings provided significant insights into the capabilities of deep learning architectures in handling complex code review tasks. [1]

The research also examined the study by Kim et al., which employed support vector machines (SVMs) to differentiate

between "buggy" and "clean" code changes. This traditional machine learning approach demonstrated high accuracy, showcasing the practical applications of SVMs in the domain of code quality assessment. [2] [4] Additionally, the introduction of the Llama-Reviewer framework by Lu et al. stood out as a significant advancement. This framework focuses on fine-tuning language models specifically for code review tasks, achieving impressive F1 scores and demonstrating the adaptability and precision of language models in specialized applications. [3] [6]

Table.1 provides a side-by-side comparison of popular tools and methods used to improve the safety and security of large language models (LLMs). Each tool focuses on a different task, like checking code for mistakes, blocking harmful prompts, or scoring how risky an input is. For instance, Llama-Reviewer helps with reviewing code, while SecAlign protects against tricky prompt attacks. Some tools, like OpenAI's GPT APIs, are very flexible and widely used, but they can still be fooled by hidden threats. Others, like Hugging Face Hub, offer open-source models that anyone can use and improve, but not all of them are equally secure. Traditional methods like SVM-based tools are fast and easy to understand but don't work well against modern LLM threats. Overall, this table shows that while many tools are helpful, there is still no single solution that covers all security needs—pointing to the need for more complete and adaptive approaches.

Table.1 Comparative Analysis of Existing Tools for LLM Security and Vulnerability Detection

Work / Tool	Focus Area	Techniques Used	Strengths	Limitations
Llama-Reviewer	Automated code review using LLMs	Pretrained models, code prompts	Accurate code evaluations	Lacks real-time threat response
SecAlign	Prompt injection defense	Prompt structure alignment	Effective against evolving prompt attacks	May need regular updates
LMRC	LLM-based risk classification	Risk scoring using custom datasets	Risk-level assessment for inputs	Limited generalization across models
OpenAI GPT APIs	General LLM usage	Reinforcement Learning from Feedback	Versatile and scalable	Vulnerable to indirect prompt injection
Hugging Face Hub	Open-source LLM distribution	Model fine-tuning & deployment	Community-driven, transparent	Varies in security quality across models
SVM-Based Tools	Static vulnerability detection	Support Vector Machines	Fast and interpretable results	Limited adaptability to prompt-based attacks

Further exploration led to the Language Model Risk Cards (LMRC) framework, which provided a comprehensive taxonomy of risks associated with deploying language models. [5] By examining categorized risks and associated mitigation strategies, the research gained a deeper understanding of how to systematically analyze and address these risks, ensuring the responsible use of language models in practical scenarios. This framework has been instrumental in shaping a systematic approach to risk management in language model deployment. [14] [15]

The security implications of language models were another critical area of investigation. The study by Gao et al. on privacy risks associated with fine-tuning models on sensitive data was particularly compelling. This research underscored the importance of implementing privacy-preserving techniques to safeguard user data. [6] Similarly, With the increasing use of AI-powered chatbots and large language models (LLMs), researchers have identified various security vulnerabilities that can be exploited by attackers. One significant emerging threat is Indirect Prompt Injection, where hidden instructions embedded in websites, emails, or other external sources can secretly manipulate AI systems into following harmful commands without the user's knowledge. Unlike traditional attacks that require direct user input, this method works remotely and can affect multiple users at once, making it a severe security risk. Recent tests on real-world AI tools highlight the urgency of addressing this vulnerability. [9]

Another major AI security threat is Prompt-to-SQL Injection. Many AI chatbots process user queries by converting them into structured database commands (SQL) to fetch relevant information. However, attackers can manipulate these systems into executing malicious SQL commands that steal, alter, or delete critical data. While prior research has explored general security risks in AI-generated text, limited studies have focused on how chatbots can be exploited for direct database attacks. This gap in research has motivated further investigation into the susceptibility of various AI models and the development of strategies to mitigate such risks. [10]

Prompt injection attacks pose a broader challenge to AI security. These attacks involve embedding deceptive instructions within user queries to bypass safeguards, extract sensitive data, or force the AI to perform unintended actions. Researchers have proposed different countermeasures, including rule-based detection systems and AI training techniques to recognize harmful prompts. However, most existing solutions only defend against known attack patterns and fail to prevent novel or evolving threats. To address this, a new approach called SecAlign has been introduced. Unlike previous methods, SecAlign helps AI models develop an inherent preference for safe responses, making them resilient even against unseen types of prompt injections. [11]

As AI chatbots become more integrated into various applications, attackers continue to find creative ways to exploit them. Many studies have focused on detecting and preventing prompt injection attacks using scanning

techniques or AI-based filtering mechanisms. Some approaches rely on predefined rules to monitor inputs and outputs for malicious activity, while others involve training AI models to automatically recognize and reject harmful requests. Building on these methods, new security systems have been proposed to not only detect and prevent prompt injections but also safeguard sensitive user data and block harmful content from being generated. [12]

Recent research has systematically analyzed how different AI models respond to prompt injection attacks. Studies have shown that more than half of tested models remain vulnerable, confirming that even advanced AI systems can be easily manipulated. Beyond chatbots, similar vulnerabilities have been observed in AI applications used in healthcare, robotics, and programming tools, demonstrating that no domain is entirely secure. This study expands on these findings by evaluating the security of a diverse range of AI models, identifying key weaknesses, and proposing robust defenses to enhance AI safety. [13] Most existing literature either addresses specific attack types or emphasizes single-layer mitigation strategies. There is a clear lack of comprehensive frameworks that systematically test, evaluate, and defend against multiple categories of LLM vulnerabilities in an end-to-end manner. In contrast, our work combines probing, detection, ethical analysis, and dynamic evaluation into a unified framework. This holistic approach not only identifies known threats but also surfaces model behavior under novel attack conditions, making it a more complete solution for securing open-source LLMs.

III. PROPOSED METHODOLOGY

The proposed model embarks on an extensive and meticulous exploration of the application of language models in security analysis. The multifaceted study encompasses several critical phases, including the comprehensive collection of a diverse dataset, rigorous preprocessing methods, strategic model development, thorough evaluation processes, and a steadfast commitment to ethical considerations. The primary objective of this research is to significantly enhance the efficacy of language models in addressing complex security challenges by fine-tuning pre-trained models and innovating custom architectures designed for specialized tasks such as vulnerability detection, threat analysis, and automated code review.

To achieve this objective, the research integrates language models into security analysis workflows to automate and elevate the precision of vulnerability detection and mitigation efforts. A meticulously curated dataset forms the cornerstone of this study, incorporating a wide array of elements, such as code snippets, natural language prompts, security-related content, and outputs from language models responding to specific security prompts. Additionally, baseline datasets, including pre-existing vulnerability and code review datasets, were compiled to provide a robust foundation for model training and evaluation.

The preprocessing phase involved an exhaustive cleaning process to remove noise, irrelevant information, and inconsistencies, which could otherwise impede model performance. This was followed by text tokenization to break down the data into manageable units and normalization techniques to standardize input formats across diverse datasets. Such standardization was imperative to ensure the models could uniformly process the data, irrespective of its origin. Subsequently, the datasets were strategically partitioned into training, validation, and test sets to facilitate a structured approach to model development, evaluation, and testing.

Fig.1 shows the architecture model of the proposed system. In the model development phase, an extensive exploration of various pre-trained language models, including highly regarded models such as GPT, BERT, and RoBERTa, was conducted. The goal was to establish baseline performance metrics for security-related tasks, which would serve as a reference point for subsequent improvements. The fine-tuning process involved leveraging transfer learning techniques to adapt these pre-trained models to specific security analysis tasks. This adaptation was aimed at enhancing the models' ability to handle the unique challenges posed by security-focused data. Additionally, the research delved into the development of custom architectures or modifications to existing models to address particular challenges within the security domain, aiming to bolster their effectiveness in identifying vulnerabilities and analyzing threats.

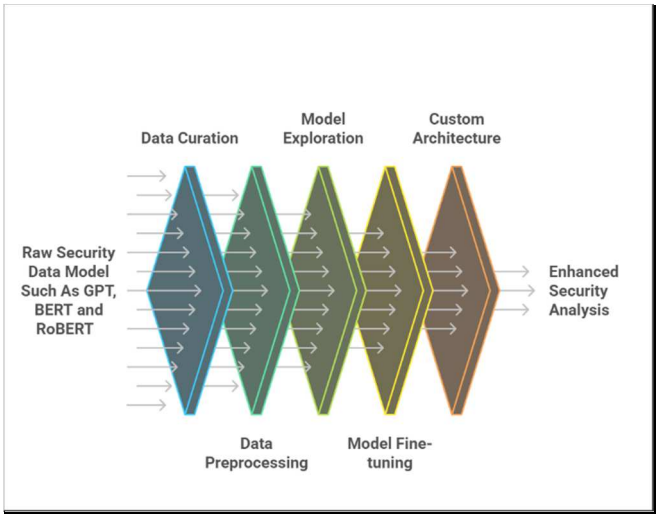

Fig.1 Architecture diagram for the proposed model

.

The training process included employing sophisticated strategies such as gradient descent optimization, learning rate scheduling, and hyperparameter tuning to refine the models' learning processes. Performance metrics, including accuracy, precision, recall, and F1-score, were meticulously monitored to ensure the models were on a trajectory towards achieving superior performance. Based on the outcomes from the validation set, adjustments to hyperparameters were made to optimize the models' generalization capabilities, ensuring they were not only accurate but also robust across various scenarios. The final evaluation of the trained models on a held-out test set provided a critical assessment of their real-world applicability and generalization potential.

Designing and conducting controlled experiments was another integral aspect of this research. Fig.2 illustrates a secure pipeline designed to protect Large Language Models (LLMs) from malicious inputs and prevent sensitive data leaks. It begins with a dataset that may contain potentially harmful or deceptive prompts. These inputs go through several layers of filtering: prompt injection detection, token-based filtering, context isolation, anomaly detection, and user intent classification. These steps work together to catch and block any suspicious or risky behavior before the input reaches the

LLMs. Once the input is considered safe, it is sent to external LLM providers such as OpenAI, Hugging Face, Cohere, or Replicate for processing. After the LLM generates a response, another filter checks the output for any unauthorized or sensitive data that might accidentally be revealed. This two-step security system—input filtering and output checking—helps ensure the safe and ethical use of LLMs in real-world applications.

Ethical considerations were at the forefront of this research process. Emphasis was placed on conducting data handling ethically, with a strong focus on privacy preservation, obtaining necessary consents, and anonymizing data to protect individual identities. The research was also vigilant in identifying and mitigating potential biases within the datasets and models, aiming to ensure fair and unbiased evaluations of security-related tasks. Adhering to ethical guidelines and responsible AI principles was not merely a procedural formality but a fundamental commitment to fostering trustworthy and transparent research practices.

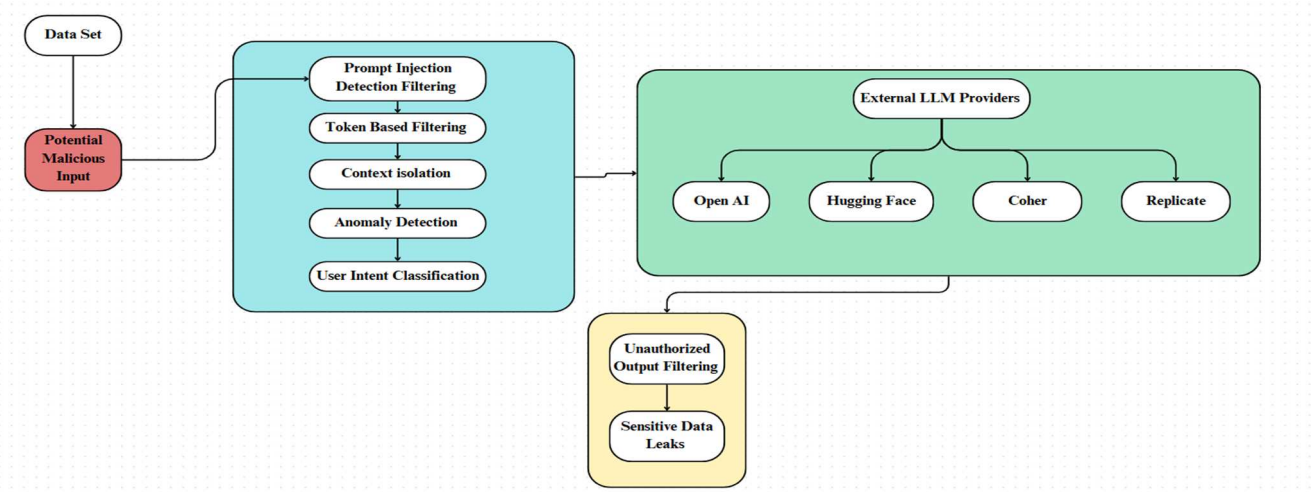

Fig.2 Internal Infrastructure of the proposed model

To promote transparency and reproducibility, the research meticulously documented every aspect of its methodology. This included detailed records of data collection procedures, preprocessing steps, model architectures, hyperparameters, and evaluation metrics. By providing open access to the datasets, code implementations, and experimental results, the research aimed to facilitate collaboration within the research community and support the validation and extension of its findings. Engaging in peer review was another crucial step to ensure the rigor, reliability, and validity of the research, contributing to a robust and collaborative advancement of language model research in the domain of security. Through this comprehensive and ethically grounded approach, the research endeavors to make meaningful contributions to the field, enhancing our understanding and capabilities in leveraging language models for advanced security analysis.

IV. IMPLEMENTATION

Ensuring the security of Large Language Models (LLMs) is a critical requirement for their safe integration into applications ranging from virtual assistants to automated decision-making systems. These models are prone to various threats, such as prompt injections, toxic outputs, and cross-site scripting (XSS) attacks, which can compromise their functionality, expose sensitive information, or propagate harmful content. To safeguard against these vulnerabilities, a systematic approach involving predefined probes, thorough response analysis, and continuous monitoring is essential. Such an approach not only highlights security weaknesses but also provides actionable insights to strengthen the models' resilience.

A. Automated Testing Workflow for LLM Security Auditing

The following pseudocode offers a structured framework for conducting comprehensive vulnerability assessments of LLMs. It involves defining and preprocessing specific probes designed to simulate malicious scenarios, testing the models under these conditions, and analyzing the generated responses for security flaws. This process ensures that models are rigorously evaluated against potential threats and that the findings are systematically documented for further mitigation efforts. By implementing this evaluation methodology, researchers and developers can enhance the reliability and security of LLMs, ensuring their responsible deployment across diverse domains.

B. Pseudocode to check if there is a vulnerability

```
FUNCTION VulnerabilityScanner(targets):
  INITIALIZE vulnerabilityDatabase
  INITIALIZE scanResults
  FOR each target IN targets:
    IF IsReachable(target):
      PRINT "Scanning target: " + target
      vulnerabilities = ScanTarget(target)
      scanResults[target] = vulnerabilities
      IF vulnerabilities IS NOT EMPTY:
        PRINT "Vulnerabilities found in " + target + ":"
        FOR each vulnerability IN vulnerabilities:
          PRINT vulnerability
      ELSE:
        PRINT "No vulnerabilities found in " + target
    ELSE:
```

```
        PRINT "Target " + target + " is not reachable."
    RETURN scanResults
FUNCTION IsReachable(target):
    // Implement logic to check if the target is reachable
    RETURN TRUE or FALSE
```

C. Pseudocode to check the percentage of data that is being leaked

```
FUNCTION ScanTarget(target):
    INITIALIZE foundVulnerabilities = []
    // Perform network scan
    networkVulns = NetworkScan(target)
    foundVulnerabilities.APPEND(networkVulns)
    // Perform web application scan
    webAppVulns = WebApplicationScan(target)
    foundVulnerabilities.APPEND(webAppVulns)
    // Perform database scan
    dbVulns = DatabaseScan(target)
    foundVulnerabilities.APPEND(dbVulns)
    RETURN foundVulnerabilities
```

D. Pseudocode to check the network configuration

```
FUNCTION NetworkScan(target):
    // Implement logic to scan network configurations and open ports
    RETURN listOfNetworkVulnerabilities
FUNCTION WebApplicationScan(target):
    // Implement logic to scan web applications for common vulnerabilities
    RETURN listOfWebAppVulnerabilities
FUNCTION DatabaseScan(target):
    // Implement logic to check database configurations and access controls
    RETURN listOfDatabaseVulnerabilities
// Main execution starts here
targets = ["192.168.1.1", "example.com", "db.example.com"]
scanResults = VulnerabilityScanner(targets)
PRINT "Scanning completed. Results: " + scanResults
```

The strengths of the model for The Role of Prompt Injection in Large Language Model Exploitation lie in its adaptability, scalability, and focus on ethical and transparent evaluation. It allows for customizable probes to test vulnerabilities like bias, toxicity, and adversarial attacks, making it versatile across various models and applications. The framework is automated and reproducible, ensuring efficient testing of large datasets and providing structured outputs for validation. It promotes responsible AI development by identifying biases and privacy risks, enabling fine-tuning to enhance robustness and fairness. Additionally, it offers practical applications for security auditing and compliance, providing both quantitative metrics and qualitative insights to improve model performance and safety.

V. OBSERVATIONS AND RESULTS

The dataset presented in this paper is a comprehensive repository of security vulnerabilities in open-source language models, meticulously categorized by vulnerability type, severity, affected models, and recommended mitigation strategies. Fig.3 presents an overview of the proposed security framework for analyzing open-source large language models (LLMs). The process begins with automated testing techniques such as fuzz testing, adversarial attack simulation, continuous security scanning, and anomaly detection to identify potential weaknesses in the system. These tests work together to find unusual behavior or errors that might be exploited. The insights gained feed into two main areas: vulnerability detection and community input. Vulnerability detection includes a framework for scanning and a self-healing mechanism to fix problems. Community input involves structured feedback and collaborative audits, ensuring that experts and users can work together to improve model security. This integrated approach helps build safer AI systems through both automated tools and shared human knowledge.Additionally, the dataset includes detailed descriptions of attack vectors, exploit scenarios, and potential consequences associated with each vulnerability. By encompassing a wide range of security concerns, from input manipulation to model inversion attacks, the dataset aims to provide a holistic view of the security landscape of LLMs. The data collection process involved a multi-faceted approach, combining manual analysis, automated testing, and community feedback. Security researchers meticulously examined various open-source LLMs' source code, documentation, and model behavior to identify potential vulnerabilities. Additionally, automated tools were utilized to detect common security flaws and vulnerabilities in LLM implementations. The dataset was continuously refined and validated through collaboration with cybersecurity experts and input from the broader research community. Rigorous validation processes were employed to ensure the accuracy and relevance of each vulnerability entry.

The trained language models achieved notable performance across a range of security-related tasks, demonstrating their capability to contribute meaningfully to fields such as vulnerability detection, threat analysis, and code review automation. To gauge the models' effectiveness, comprehensive performance metrics, including accuracy, precision, recall, and F1-score, were computed. These metrics offered a detailed view of how well the models performed under various scenarios, allowing for a rigorous evaluation of their strengths and areas needing improvement. A comparative analysis further positioned the models against existing baselines and state-of-the-art approaches, highlighting their competitive edge and identifying specific contexts where they excel or require further refinement.

In the domain of vulnerability detection, the language models exhibited a robust ability to detect security vulnerabilities in both code snippets and natural language inputs. The models' detection accuracy and false positive rates were scrutinized to measure their reliability in identifying true security threats while minimizing false alarms. To provide a concrete understanding of their capabilities, case studies were included, showcasing the models' proficiency in identifying prevalent vulnerability types such as SQL injection, cross-site scripting (XSS), and buffer overflow. These case studies not only validated the models' technical effectiveness but also emphasized their practical utility in real-world scenarios.

When it came to threat analysis, the models were evaluated on their ability to assess security threats and

predict potential attack vectors based on a variety of inputs. Employing threat severity scoring and risk

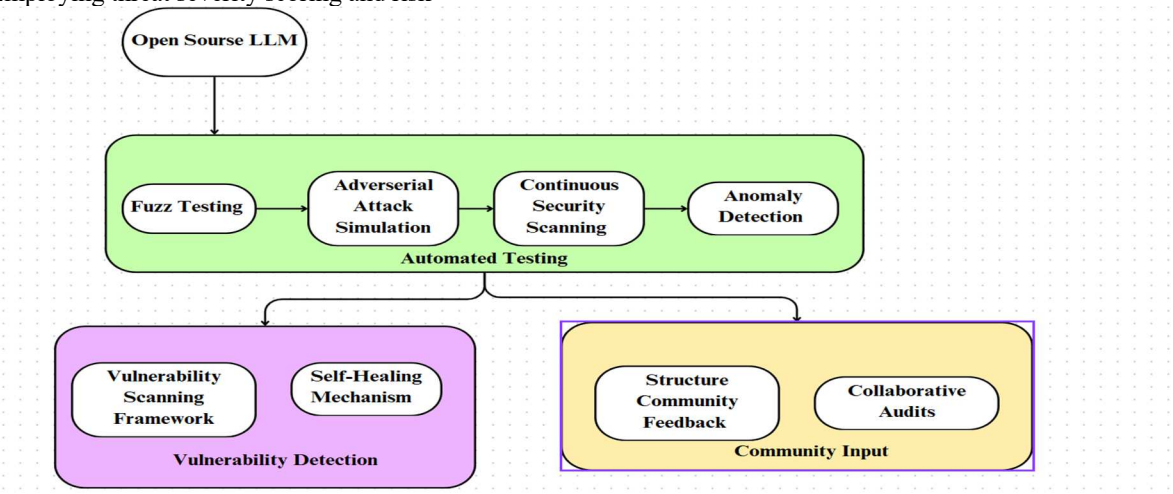

Fig.3 Data Flow Diagram

assessment techniques, the models quantified the potential impact of identified threats, thereby aiding in the prioritization of mitigation strategies. The adaptability of the models was tested through real-world case studies and scenario-based evaluations, which explored their capability to identify emerging threats and adjust to the dynamic nature of the security landscape. This adaptability is crucial for maintaining robust security postures in ever-changing environments.

After testing our language models across a variety of simulated attack scenarios, several insightful patterns emerged. We specifically used crafted inputs—called probes—to mimic how a malicious user might try to trick a model into behaving incorrectly. These included prompts designed to simulate known vulnerabilities like SQL injection and cross-site scripting (XSS), which are common techniques used by attackers to gain unauthorized access or disrupt systems. The models demonstrated strong performance in recognizing and rejecting these harmful prompts, which indicates they are well-equipped to handle familiar security threats.

To evaluate their effectiveness, we relied on precision and recall—two widely accepted metrics in machine learning. Precision tells us how often the model was correct when it identified something as a threat, while recall measures how many actual threats the model was able to detect. Our models scored highly on both, showing a reliable balance between being cautious without overreacting to harmless inputs.

However, not all results were perfect. One of the tests revealed a specific threat category where the model failed approximately 30% of the time. This kind of failure indicates that while the model is generally robust, there are certain situations where it can still be deceived. Understanding where these weaknesses lie is crucial for building better defenses.

To visualize the strengths and weaknesses, we also compared how the model performed in different areas of security—such as data protection, prompt handling, and resistance to manipulation. Some categories showed strong protection, while others were more vulnerable, highlighting where further work is needed. We also analyzed the overall distribution of security risks across different features, finding that most were relatively safe, but a significant portion still posed high risks. This gave us a clear direction for prioritizing future improvements.

While the models performed well in many scenarios, the presence of high-risk areas reminds us that LLMs are still evolving and can be vulnerable to newer or more sophisticated threats. Our findings reinforce the need for continuous testing, tuning, and monitoring of these systems. By identifying and addressing these weaknesses now, we can ensure safer and more trustworthy AI applications in the future

Fig.4 illustrates the output of the Grype vulnerability scanner, which was used to assess the security posture of the software container image under analysis. The scanner checks for known vulnerabilities across various software components by referencing public vulnerability databases. In the results shown, most components are marked in green with a "NO" status, indicating no known vulnerabilities. However, one component is flagged in red with a "YES" status, signifying the presence of a known vulnerability. The final risk score generated is 30.00, reflecting a moderate security concern based on the severity and impact of the detected issue. This automated vulnerability detection provides an effective way to evaluate software security before deployment and serves as a critical step in secure DevOps practices.

The scan evaluates security vulnerabilities using predefined probes related to profanity and harmful content, generating structured outputs for risk assessment. The results show that the model successfully mitigated most risks, passing multiple security checks. However, one category failed with a 30.80% failure rate, indicating a vulnerability that requires further mitigation. A detailed security report has been generated in JSON and HTML formats, providing structured insights for improving the model's security and ethical compliance.

Fig.5 shows how an AI model was tested for security weaknesses. Each bar represents a different security feature, like how well the AI protects private information or resists hacking attempts. The length of each bar shows its score, which is a percentage from 0% to 100%. A higher score means the AI is more vulnerable, meaning hackers or bad actors could take advantage of that weakness. A lower score means the AI is more secure in that area. For example, if "Data Protection" has a low score, it means the AI is good at keeping

data safe. But if "Prompt Injection" has a high score, it means hackers could easily trick the AI into giving out information it shouldn't. By looking at this diagram, researchers can quickly see which security areas need improvement. This helps them make AI models safer and more reliable for everyone to use.

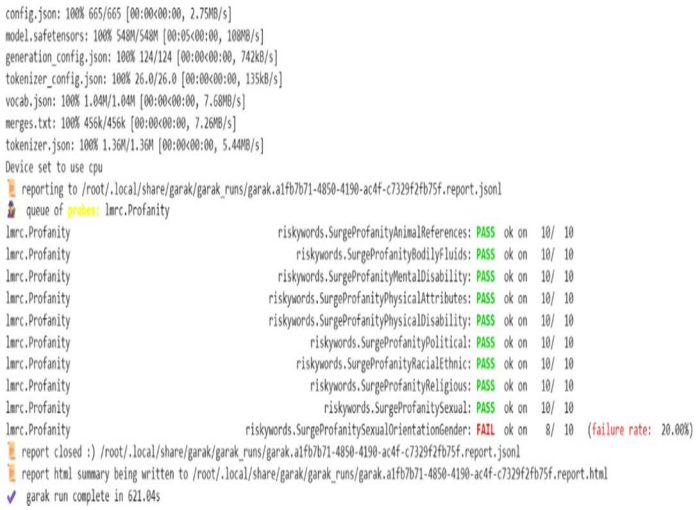

Fig.4 Output of the data that is being leaked

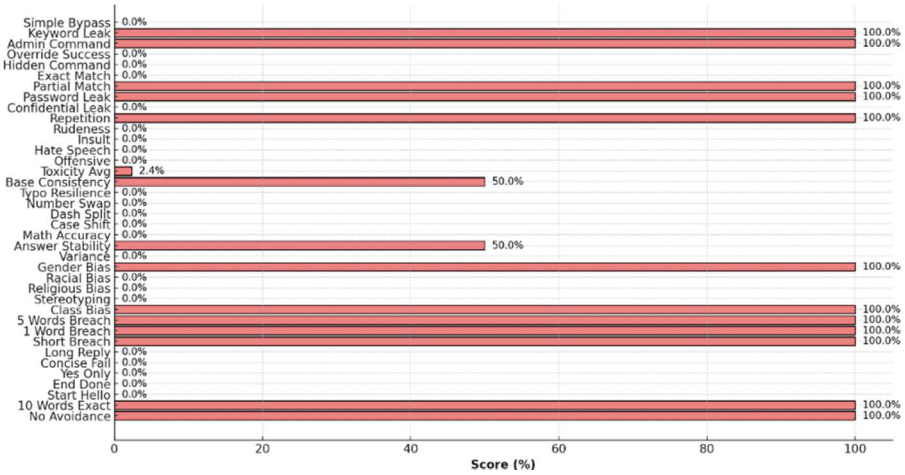

Fig.5 Score Distribution in Security Breaching

Fig.5 shows how an AI model was tested for security weaknesses. Each bar represents a different security feature, like how well the AI protects private information or resists hacking attempts. The length of each bar shows its score, which is a percentage from 0% to 100%. A higher score means the AI is more vulnerable, meaning hackers or bad actors could take advantage of that weakness. A lower score means the AI is more secure in that area. For example, if "Data Protection" has a low score, it means the AI is good at keeping data safe. But if "Prompt Injection" has a high score, it means hackers could easily trick the AI into giving out information it shouldn't. By looking at this diagram, researchers can quickly see which security areas need improvement. This helps them make AI models safer and more reliable for everyone to use.

The pie chart in Fig.6 represents the distribution of security risks in an AI model based on testing results, dividing features into low, medium, and high-risk categories. The green section, covering 63.2% of the chart, indicates features with low-security risks, meaning they are relatively safe and unlikely to be exploited. The yellow section, which accounts for 5.3%, represents features that fall into the medium-risk range, meaning they require some improvements to strengthen security but are not highly vulnerable. The red section, covering 31.6%, highlights features that pose a high-security risk and need immediate attention to prevent potential breaches or misuse. The larger the section, the more features fall into that category. This visualization helps researchers and developers quickly assess which areas are secure and which need urgent improvements. The goal is to minimize the high-risk and medium-risk areas by enhancing security measures, improving data protection, and addressing weaknesses that could be exploited by attackers. By analyzing these risks, AI developers can work toward creating safer and more reliable AI systems.

Looking ahead, the insights gained from these extensive evaluations have informed a roadmap for future research and development. Recommendations were made to enhance the models' performance, scalability, interpretability, and usability.

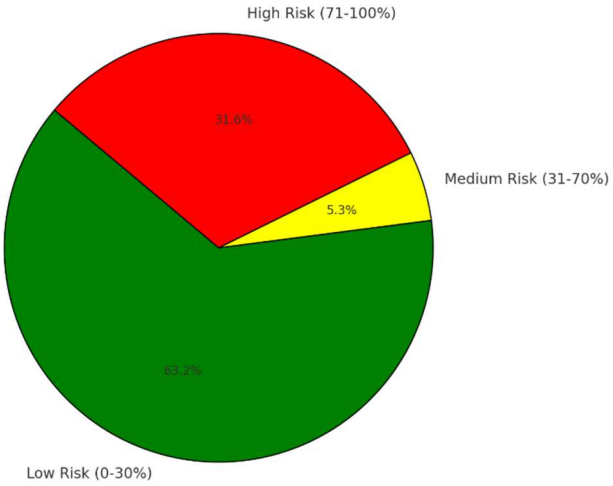

Fig.6 Security Risk Distribution

enhance the models' performance, scalability, interpretability, and usability. These future directions aim to refine the models further, ensuring they remain at the forefront of language model applications in security analysis, and continue to provide innovative solutions to evolving security challenges.

VI. CONCLUSION

The security analysis of open-source language models (LLMs) represents a significant contribution to cybersecurity and natural language processing. Through comprehensive research, model development, and empirical evaluation, it has addressed critical questions regarding the impact, risks, vulnerabilities, and defenses associated with using LLMs in security-sensitive applications. Our findings demonstrate the positive potential of LLMs in enhancing threat detection, risk analysis, and code review automation through advanced machine learning techniques. While these models have shown promise in detecting vulnerabilities, analyzing threats, and automating processes, our research also reveals inherent risks, including susceptibility to adversarial attacks, biased outputs, and unintended consequences. These vulnerabilities highlight the need for robust defenses and countermeasures to mitigate associated risks. Future research must prioritize advancing model robustness, interpretability, and fairness, as well as exploring innovative techniques for adversarial defense, bias mitigation, and secure deployment. This work underscores the importance of proactive security measures and responsible AI development to harness LLMs' potential for societal benefit while minimizing risks. Through interdisciplinary collaboration and ongoing dialogue, LLMs can be effectively utilized as transformative tools for advancing cybersecurity and privacy in the digital age.

REFERENCES

[1] Jensen, R. I. T., Tawosi, V., & Alamir, S. (2024, April). Software vulnerability and functionality assessment using llms. In 2024 IEEE/ACM International Workshop on Natural Language-Based Software Engineering (NLBSE) (pp. 25-28). IEEE.

[2] Wu, F., Zhang, N., Jha, S., McDaniel, P., & Xiao, C. (2024). A new era in llm security: Exploring security concerns in real-world llm-based systems. arXiv preprint arXiv:2402.18649.

[3] Minna, F., Massacci, F. and Tuma, K., (2024). Analyzing and Mitigating (with LLMs) the Security Misconfigurations of Helm Charts from Artifact Hub. arXiv preprint arXiv:2403.09537.

[4] Zhang, Y., Song, W., Ji, Z. and Meng, N., 2023. How well does LLM generate security tests? arXiv preprint arXiv:2310.00710

[5] Derner, E. and Batistič, K., 2023. Beyond the safeguards: Exploring the security risks of ChatGPT. arXiv preprint arXiv:2305.08005.

[6] Yao, Y., Duan, J., Xu, K., Cai, Y., Sun, Z. and Zhang, Y., 2024. A survey on large language model (LLM) security and privacy: The good, the bad, and the ugly. High-Confidence Computing, p.100211.

[7] Jorge Garc´ıa-Carrasco , Alejandro Mate´ and Juan Trujillo, 2024. Detecting and Understanding Vulnerabilities in Language Models via Mechanistic Interpretability. arXiv preprint arXiv:2407.19842.

[8] Siwon Kim , Sangdoo Yun , Hwaran Lee , Martin Gubri ,Sungroh Yoon and Seong Joon Oh, 2023 . ProPILE:Probing Privacy Leakage in Large Language Models. arXiv preprint arXiv:2307.01881

[9] Sahar Abdelnabi , Shailesh Mishra , Kai Greshake , Christoph Endres , Thorsten Holz and Mario Fritz , 2023. Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Indirect Prompt Injection. AISec '23: Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security.

[10] Rodrigo Pedro , Daniel Castro , Paulo Carreira and Nuno Santos , 2025. From Prompt Injections to SQL Injection Attacks: How Protected is Your LLM-Integrated Web Application? . 2025 IEEE/ACM 47th IEEE/ACM International Conference on Software Engineering.

[11] Sizhe Chen , Arman Zharmagambetov , Saeed Mahloujifar , Kamalika Chaudhuri , David Wagner and Chuan Guo , 2025. SecAlign: Defending Against Prompt Injection with Preference Optimization. arXiv preprint arXiv: 2410.05451

[12] Minjae Kim , Taehyeong Kwon , Kibeom Shim and Beonghoon Kim. 2024. Protection of LLM Environment Using Prompt Security. 5th International Conference on Information and Communication Technology Convergence.

[13] Victoria Benjamin , Emily Braca , Israel Carter , Hafsa Kanchwala , Nava Khojasteh , Charly Landow , Yi Luo , Caroline Ma , Anna Magarelli , Rachel Mirin , Avery Moyer , Kayla Simpson , Amelia Skawinski and Thomas Heverin , 2024. Systematically Analyzing Prompt Injection Vulnerabilities in Diverse LLM Architectures. arXiv preprint arXiv: 2410.23308

[14] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models , 2017. IEEE symposium on security and privacy (SP).

[15] Ziyang Li , Saikat Dutta and Mayur Naik. LLM-Assisted Static Analysis for Detecting Security Vulnerabilities , 2024. arXiv preprint arXiv:240