Lifting the Veil on the Large Language Model Supply Chain: Composition, Risks, and Mitigations

Kaifeng Huang
Tongji University
Shanghai, China

Bihuan Chen
Fudan University
Shanghai, China

You Lu
Fudan University
Shanghai, China

Susheng Wu
Fudan University
Shanghai, China

Dingji Wang
Fudan University
Shanghai, China

Yiheng Huang
Fudan University
Shanghai, China

Haowen Jiang
Fudan University
Shanghai, China

Zhuotong Zhou
Fudan University
Shanghai, China

Junming Cao
Fudan University
Shanghai, China

Xin Peng
Fudan University
Shanghai, China

Abstract

Large language models (LLM) have sparked significant impact with regard to both intelligence and productivity. In recent years, a great surge has been witnessed in the introduction of both commercial and open-source LLMs. Many businesses have adopted the LLMs into their applications to solve their own domain-specific tasks. However, integrating LLMs into specific business scenarios requires more than just utilizing the models themselves. Instead, it is a systematic process that involves substantial components, which are collectively referred to as the LLM supply chain. The LLM supply chain inherently carries risks. Therefore, it is essential to understand the types of components that may be introduced into the supply chain and the associated risks, enabling different stakeholders to implement effective mitigation measures. While some literature discusses risks associated with LLMs, there is currently no paper that clearly outlines the LLM supply chain from the perspective of both providing and consuming its components. As LLMs have become essential infrastructure in the new era, we believe that a thorough review of the LLM supply chain, along with its inherent risks and mitigation strategies, would be valuable for industry practitioners to avoid potential damages and losses, and enlightening for academic researchers to rethink existing approaches and explore new avenues of research. Our paper provides a comprehensive overview of the LLM supply chain, detailing the stakeholders, composing artifacts, and the supplying types. We developed taxonomies of risk types, risky actions, and mitigations related to various supply chain stakeholders and components. In summary, our work explores the technical and operational aspects of the LLM supply chain, offering valuable insights for researchers and engineers in the evolving LLM landscape.

1 Introduction

The large language models (LLMs) have sparked unprecedented discussion about the power of generative language models, showcasing their remarkable ability to understand human-written text, embody vast knowledge, and generate responses based on user instructions. In 2023 following ChatGPT's debut, there has been a significant surge in the development and introduction of large language models, regarding both commercial LLMs and free open-source LLMs. Researchers and practitioners have extended the capabilities of LLMs beyond natural language processing, applying them in fields like software engineering, finance, and education, dramatically reshaping these areas [7, 29, 89].

Integrating LLMs involves more than just utilizing the models themselves. On one hand, although some applications may directly employ commercial LLMs, they still require extra solutions for pre-processing (*e.g.,* Promptify [34] for prompt engineering, and GPT-Cache [33] for reducing LLM API call expenses), post-processing (*e.g.,* filtering out unwanted content), integration (*e.g.,* integrating with websites, servers, or mobile apps), and plugin interactions (*e.g.,* Zapier [35] for linking with external apps). On the other hand, businesses may prefer open-source LLMs to enable flexible customizations, which demand more sophisticated components, during stages such as pre-training, model reusing, fine-tuning (*e.g.,* Labelbox [55]), model conversion (*e.g.,* Onnx [75]), quantization (*e.g.,* TensorRT [74]) and deployment. Behind the directly used components, there exists a multitude of sub-components that are transitively depended upon, further increasing the overall dependency complexity. Furthermore, the supplying types in the LLM supply chain are more varied than in traditional software supply chains. Data and models introduce additional supply types, including partial data reuse, model parameter reuse through data augmentation, fine-tuning, etc. Overall, leveraging LLMs is a systematic process that entails numerous components and supplying types across multiple stages, collectively referred to as the LLM supply chain.

Accordingly, the risks in the LLM supply chain have been significantly amplified due to an expanded attack surface and a growing number of participants. For stakeholders, it is essential to understand their providers and clients, how their artifacts are served, the risks associated with them, and the mitigation measures that are required. While the LLM supply chain shares similarities with

the well-recognized open-source software supply chain [56], it also exhibits its uniqueness. For instance, attackers can exploit publicly accessible LLMs through prompt engineering to extract sensitive information (*e.g.,* privacy leakage) or fine-tune open-source LLMs [27] to embed malicious instructions within the model (*e.g.,* back-door attacks). Therefore, a comprehensive understanding of the types of components and supplying types in the LLM supply chain, thus to understand their corresponding risks is crucial to enable stakeholders to implement effective mitigation strategies.

While some literature discusses risks associated with LLMs, there is currently no paper that clearly outlines the LLM supply chain from the perspective of both providing and consuming its components. As a result, the inherent risks and potential mitigation strategies are less organized and comprehensive. Ladisa et al. [56] developed taxonomies for attack and defense techniques in software supply chains. However, their taxonomies do not account for LLM-specific components such as pre-trained LLMs, prompts, and data, making their survey less attractive for LLM supply chain stakeholders. Several surveys address the security and privacy challenges [19, 26, 54, 70, 121, 121] and discuss the defense and mitigation strategies of LLMs. However, they did not explore which stakeholders and artifacts are affected, which may limit the ability to provide actionable insights and a clear scope for the relevant stakeholders. Additionally, Wang et al. [112] were among the first to outline a research agenda for the LLM supply chain. They highlighted the key components in the LLM supply chain, illustrating them through the interactions within MLOps and DevOps. As the topic is still emerging and rapidly evolving, existing works may not comprehensively incorporate insightful ideas and perspectives from online preprints or web blogs. Therefore, we believe that a comprehensive LLM supply chain overview as well as a taxonomy of risks and clear guidance of mitigation in the LLM supply chain could be beneficial for both industrial and academic audiences. It could guide practitioners to pay attention to the *LLM supply chain* risks and provide a visionary map for academic researchers to devise new mitigation or defensive techniques.

In our paper, we provide a comprehensive overview of the LLM supply chain, detailing the stakeholders, composing artifacts, and the supplying types, as illustrated in Figure 1. Our definition of the LLM supply chain differs from Want et al.'s work [112] in emphasizing the LLM supply chain more on the basic components and their supply relationships between them. We developed taxonomies of risk types, risky actions, and mitigations related to various LLM supply chain stakeholders and components. Specifically, we fist holistically collected both academic and online resources. Then, we summarize the risk scenarios as *stakeholders* performing *risky actions* on specific *supply chain components*, which in turn lead to distinct *risk types*. We illustrate the risk scenarios from the perspective of upstream contributors, downstream users, and administrators. Furthermore, we list the types of various mitigation measures in response to the LLM supply chain risks.

Our paper explores the technical and operational aspects of the LLM supply chain, offering a holistic understanding that draws valuable insights for researchers and engineers in software engineering, system architecture, software security, and data governance. The main contributions of our paper are as follows:

- We presented a comprehensive overview of the LLM supply chain, including artifacts, stakeholders, and supplying types.
- We develop a detailed taxonomy of risk and mitigations with regard to stakeholders, risk types and supply chain artifacts, providing actionable guidance for practitioners involved in the LLM supply chain.
- We envision future challenges and opportunities in securing the LLM supply chain.

2 Large Language Model Supply Chain

We first provide our definition for the large language model supply chain. Then, we detail the stakeholders and composing components involved in the LLM supply chain, and provide a description of the LLM development stages. Lastly, we compare our work with existing surveys relevant to the LLM supply chain.

2.1 Definition

We present an overview of the large language model supply chain in Figure 1. While it may not capture the entire landscape exhaustively, we have aimed to make it as comprehensive as possible. The large language model (LLM) supply chain encompasses the ecosystem and sequence of processes integral to developing, training, deploying, and distributing applications that leverage large language models. This supply chain includes:

- *Stakeholders in various roles.* Stakeholders take different roles based on the specific supply artifact and action. For instance, they may include developers contributing to the artifacts, organizations managing the platforms, or vendors providing cloud computing services.
- *Upstream artifacts in multiple forms.* Upstream artifacts are those on which downstream applications have dependencies on. For example, these artifacts may include plaintext source code, binary executables, or serialized models.
- *Diverse supply relationships.* As artifacts vary, the types of supply relationships differ. For instance, they may include augmentation relationships for data, cloning relationships for code, and merging relationships for models.
- *Development stages for various purposes.* The development stages include a comprehensive scope for LLM application. *e.g.,* data pre-processing, model pre-training, fine-tuning, integration, delivery, monitoring, and feedback.

2.2 Composing Components

We summarize five components in the LLM supply chain, i.e., *stakeholder*, *artifact*, *toolchain*, *platform*, and LLM applications (*apps*).

2.2.1 Stakeholders. We summarize four roles of stakeholders in the LLM supply chain, i.e., *contributors*, *consumers*, *administrators*, and *user*. Note that we categorize the stakeholders as roles because an individual can occupy a single role or assume multiple roles simultaneously, such as being both an *contributor* and an *administrator* in an open-source package platform.

Contributors. We collectively refer to contributors engaged in development activities and sharing artifacts, such as data, models, prompts, or third-party libraries, as *contributors*. Their roles vary depending on the types of artifacts they contribute.
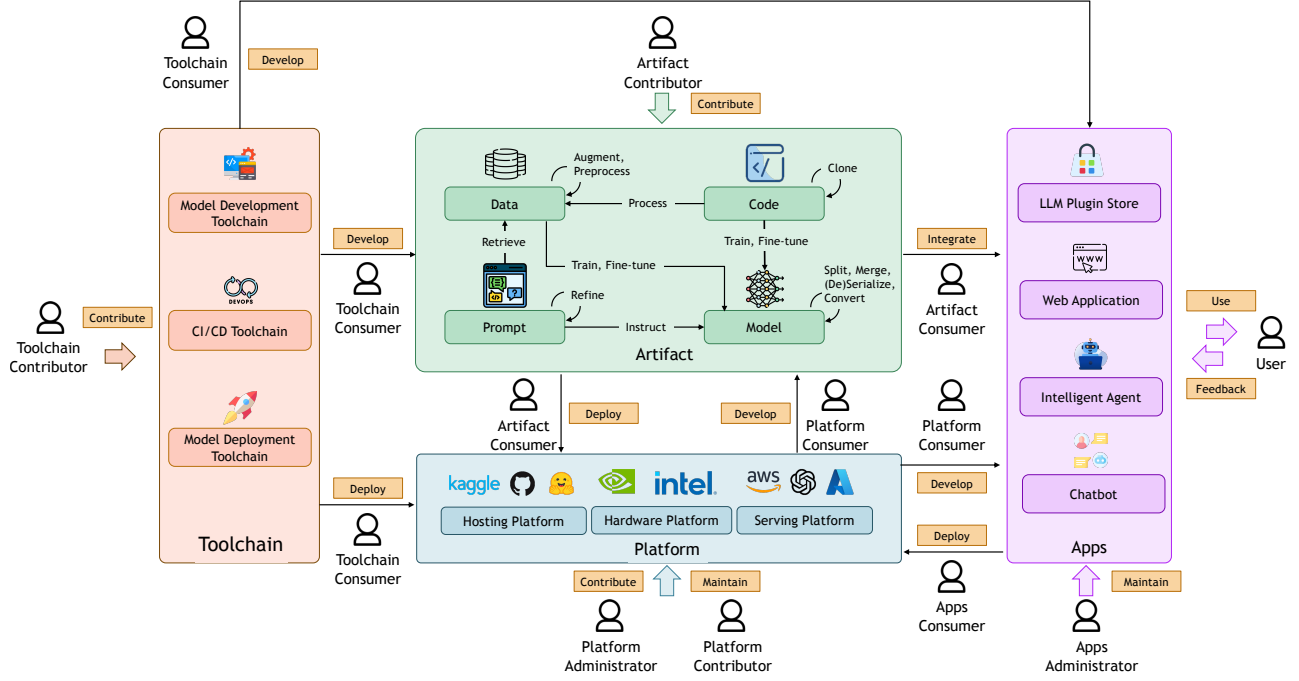
Figure 1: Overview of the Large Language Model Supply Chain

- *Platform Contributors* are responsible for contributing to hosting platforms (*e.g.,* hosting artifacts), hardware platforms, and serving platforms (*i.e.,* Google Cloud [18], Amazon Web Services [91]). They are obligated to provided secure, stable, isolated, and managed environments. *i.e.,* malicious content detection, safe system images, authorized cloud accounts, etc.
- *Artifact Contributors* collect, process, and upload various forms of data, code, prompts, and models to the hosting platforms. They can be data engineers, model developers, prompt engineers, and library developers.
- *Toolchain Contributors* are responsible for developing, maintaining, and improving the tools used throughout the model lifecycle. *e.g.,* Jupyter, TensorFlow for model development, Jenkins, GitHub Actions in CI/CD.They ensure that these tools are reliable, scalable, and compatible with the requirements from *Toolchain Consumers*.

Consumers. We distinguish consumers depending on their consuming components, including platform consumers, artifact consumers, toolchain consumers, and app consumers.

- *Platform Consumers* are entities that use the platform infrastructure for various purposes, such as accessing hosting services, utilizing computational resources, or interacting with serving platforms for deploying and running models and applications.
- *Artifact Consumers* are those who utilize the artifacts produced and shared within the platform. This includes data sets, code, prompts, and models.
- *Toolchain Consumers* are individuals or teams that leverage the toolchain components for developing, deploying, and maintaining models. They use model development, CI/CD, and deployment tools to streamline the lifecycle of models and applications.

- *App Consumers* are end-users or other systems that interact with applications built using the platform's resources and artifacts.

Users. The users of LLM applications perform interactions with LLM applications in various forms. Their feedbacks (explicitly or implicitly) may be collected back to the LLM application.

Administrators. We summarize administrators as platform administrators and apps administrators.

- *Platform Administrator* include two sub roles. The platform administrators manage and oversee the infrastructure, ensuring the system's operational integrity, security, and performance. The project administrators, on the other hand, focus on managing repositories of artifacts, such as data repositories, model repositories, prompt repositories/websites, and library repositories. They hold the privileges to merge pull requests, manage versioning, and onboard new contributors.
- *Apps Administrator* is responsible for managing the lifecycle of applications that rely on the artifacts and models within the platform. This includes deploying, configuring, and monitoring applications.

2.2.2 Artifacts. The LLM supply chain artifacts include data, model, prompts and code.

Data. Data serves as the foundational element in the workflow. It can be augmented and preprocessed from the original data to improve its quality, ensuring it is ready for use in training and fine-tuning the model. Retrieval methods bring in relevant data from various sources, facilitating the prompt generation process. They can be used as the data for pre-training or fine-tuning, or as the knowledge database for retrieval augmented generation.

Model. The model represents the large language model that is trained and fine-tuned using the processed data and instructions

from the prompt. The model may be adjusted or modified by splitting, merging, serializing, or deserializing components as needed to optimize performance. It generates predictions or outputs based on the prompts used as the instructs.

Prompt. The prompt is crafted to instruct the model effectively, guiding it on what information to retrieve, process, or generate. Prompts can be refined to adapt to specific scenarios, leading to more accurate or tailored instructions that improve model outputs.

Code. The code encompasses the software and scripts utilized for data processing, model training, and prompt refinement. This includes code that directly invokes relevant library APIs, as well as the libraries themselves as dependencies. The code can be cloned for customization to enhance compatibility and functionality.

2.2.3 Platforms. The platforms are publicly accessible websites that provide essential infrastructure and services for distributing, and deploying various components of the LLM ecosystem.

Hosting Platform. These platforms facilitate the storage, sharing, and versioning of software packages, models, data, and related resources required for LLM development and deployment. *e.g.,* Data Hosting Platforms (*e.g.,* Kaggle [63] for datasets), Model Hosting Platforms (*e.g.,* Hugging Face [28] for pre-trained models), Repository Hosting Platforms (*e.g.,* GitHub [32] for code repositories), Package Registries (*e.g.,* PyPI [30] for Python packages, Debian [83] for Linux packages), Plugin Marketplaces (e.g., Visual Studio Marketplace [67] for development tools), and Prompt Sharing Platforms where users can share and discover prompt engineering techniques.
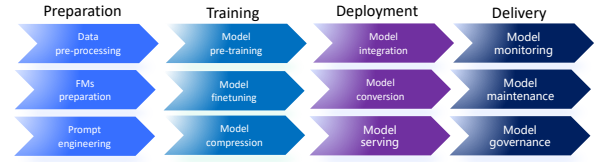
Hardware Platform. These platforms consist of specialized hardware designed to optimize the performance of LLMs. They include dedicated chips and accelerators (*e.g.,* NVIDIA GPUs, Google TPUs) that provide the computational power necessary to handle large datasets and complex models, reducing training time and improving efficiency in deployment.

Serving Platform. These platforms offer cloud-based services to support the training, deployment, and maintenance of LLMs. They include Cloud Computing Platforms that provide scalable computing resources, such as Google Cloud [18] and Amazon Web Services [91]. These platforms allow users to leverage high-performance infrastructure, including GPUs and TPUs, for intensive tasks like model training and inference.

2.2.4 Toolchain. We discuss the relevant toolchain with regard to model development toolchain, model deployment toolchain, and CI/CD toolchain.

Model Development Toolchain. Tools used for training or fine-tuning the LLMs. If an attacker implants malicious code in these tools, it could lead to corrupted models or abnormal outputs. *Environment Image* is the base environment (*e.g.,* Docker containers) in which the model runs could be compromised, affecting system security or performance. *Model Serializer and Deserializer* save or load models. If compromised, they could modify the model's internal state or introduce backdoors.

Model Deployment Toolchain. Tools specifically used for creating and deploying LLMs. Any manipulation or vulnerabilities here could compromise the model's behavior or safety. *Model Quantization Tool* reduce the size of the deployment model, making it more efficient. If compromised, these tools could degrade model performance or introduce malicious behavior. *Model Conversion*

Figure 2: Large Language Model Development Stages

Tool converts models between formats or versions. A poisoned conversion process could lead to abnormal model behavior or failures.

CI/CD Toolchain. Continuous Integration/Continuous Deployment systems that automate the software delivery process. A compromised CI/CD pipeline could introduce vulnerabilities during model updates.

2.2.5 Applications. Applications provide functionalities and services directly used by users. These applications include chatbots, web applications, intelligent agents, text-generation tools, and more. Application developers integrate large models into real-world products to develop intelligent applications with API interfaces. For example, the LLM plugin store can offer a wide range of customizable plugins that allow developers to easily access and implement various functionalities tailored to specific needs. This ecosystem is examined in detail in Zhao et al.'s forward-looking analysis of the LLM app store [128].

2.3 Model Development Lifecycle

Figure 2 illustrates the four essential stages of the large language model (LLM) lifecycle. Each stage includes specific processes crucial to building, deploying, and maintaining LLMs effectively. In the *Preparation stage*, foundational work is completed, including data pre-processing to clean and organize raw data, foundation model (FM) preparation to set up initial models, and prompt engineering to guide model outputs effectively. These steps ensure that the inputs are optimized for subsequent stages. The *Training stage* focuses on developing the model's capabilities. Here, model pre-training introduces the model to large datasets, enabling it to understand language patterns. This is followed by fine-tuning on specific tasks to enhance accuracy and model compression to optimize the model for efficient deployment. In the *Deployment stage*, the trained model is integrated into real-world applications. This involves model integration to embed the LLM within applications, model conversion to make it compatible with different systems, and model serving to set up scalable infrastructure for user interactions. Finally, the *Delivery stage* ensures the ongoing performance and governance of the model. Model monitoring is conducted to track performance and detect issues, model maintenance includes regular updates and retraining to retain effectiveness, and model governance oversees the ethical and regulatory aspects, ensuring compliance and responsible usage.

2.4 Related Surveys

We conducted a survey of the security aspects of large language models (LLMs). Specifically, we gathered 12 surveys from our literature collection (see Section 3.1) that discuss various security-related topics, including LLM risks, privacy challenges, adversarial attacks,

and defenses. For ease of comparison, we categorized the topics into two main areas: *attacks and risks* and *defensive techniques*. Additionally, we unified the terminology and definitions across the surveys to enable a clearer comparison.

Attack and Risks. In the context of attacks and risks, the *Output Risk* is extensively covered in eight survey papers, which refers to the potential for LLMs to generate harmful, untruthful, hallucinatory, or unhelpful content. Following this, *Privacy Attacks* are another major concern, with nine papers addressing various aspects such as membership inference attacks, privacy leakage (e.g., PPI) attacks, gradient leakage attacks, model inversion attacks, and attribute inference attacks. Besides, *Prompts Attack* is covered in 7 papers, including prompt injection attack and jailbreaking attack. Unfortunately, only a few papers [19, 81] mention *Toolchain Attacks*. Yao et al. [121] discuss *Supply Chain Vulnerabilities*, and Wu et al. [116] highlight issues like malicious webtool misuse and cross-session access, which are closely related but account for a subset of *Toolchain Attacks*. Therefore, the topic of toolchain attacks was not explored in depth. Finally, only Neel et al. [70] addressed the issue of *Copyright Risks*.

Defenses and Mitigation. Defenses and Mitigation are suggested in accordance with the corresponding attacks and risks. For *Prompt Attacks*, existing works propose *Input Sanitization* [46, 81]. To mitigate *Output Risk*, approaches like *Output Sanitization* are recommended by several studies [19, 54, 121]. For *Data Attacks*, the literature suggests techniques such as *Data Cleaning* [121], *Data De-duplication* [70], and *Data Sanitization* [26]. However, these defensive techniques are not well-summarized across the surveyed works. While some methods, such as *Differential Privacy*, *Federated Learning*, or *LLM Alignment*, refer to specific technical measures designed to defend against particular attacks or mitigate associated risks, other defensive measures are less detailed. For instance, strategies like *Output Detection* and *Input Sanitization* are suggested to address *Output Risk* and *Prompt Attacks*, respectively, but they lack in-depth technical details on how these defenses are implemented.

Comparison to Existing Surveys. Our paper contributes the following contributions which are novel and incremental to existing surveys. First, we conducted a comprehensive survey in both academic literature and online resources regarding the *open-source LLM supply chain*. Second, we provide a comprehensive overview and highlight the key stakeholders and components in the LLM supply chain. Third, we summarize the risk scenarios as *stakeholders* performing *risk actions* on specific *supply chain components*, as well as *risk types*. We illustrate the risk scenarios from the perspective of three major stakeholders. Furthermore, we list the types of various mitigation measures in response to the LLM supply chain risks.

3 Methodology

In this section, we elaborate the methodology for our study. The overview of our methodology is presented in Figure 3.

3.1 Literature Collection

We focused our literature collection on academic literature and online resources. It helps to cover a comprehensive scope for our study. On the one hand, academic literature provides a rigorous understanding of the risks and mitigations in the LLM supply chain.
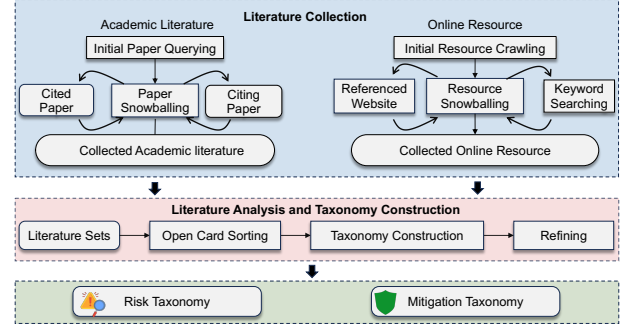

Figure 3: Approach Overview of our Literature Review

On the other hand, online resources provide real-time, validated observations and practical examples that reflect current real-world practices.

3.1.1 Academic Literature Collection. The academic literature collection consists of two phases. First, we collected the initial papers set by querying with keywords that are central to our topic. Second, we expanded the paper set using the snowballing technique, incorporating both the references cited by the initial set and the papers that cite them.

Initial Paper Set. Two PhD students with over five years of experience in software supply chain management and two PhD students with over three years in LLM security have engaged in the process. Specifically, they first collected keywords from three domains. *i.e.,* supply chain security (*e.g.,* security, attack, malware, threat, vulnerabilities), LLM security (*e.g.,* data poisoning, data provenance, model poisoning) and LLM DevOps (*e.g.,* data validation, model validation, model evaluation). The remaining participants cross-checked the keywords, and together they compiled the final comprehensive set. Afterwards, we constructed queries based on the following rules using the advanced search interface on Google Scholar.

```
(LLM security AND supply chain security)
OR (LLM security AND LLM DevOps) OR
(supply chain security AND LLM DevOps)
OR LLM security
```

Furthermore, we excluded papers that do not align with the goal of our study. First, we filter out papers published before 2018. Second, we exclude short papers with fewer than 2 pages. Finally, we manually review the remaining papers to assess their relevance to the large language model supply chain. Consequently, this process leaves us with 127 papers.

Snowballed Paper Set. We identified cited and citing papers from the initial set, with each new group of relevant papers forming the basis for the next round of reviews. This iterative process continued until we reached a point where no new relevant papers were found. Ultimately, this process led us to a final set of 313 papers.

3.1.2 Online Resource Collection. In addition to collecting scientific literature, we have also gathered online resources which often include real-world attacks or the latest knowledge not yet covered by scientific literature.

Table 1 lists the websites where we collected online resources. We chose these websites for their wide coverage, reputation in the field

Table 1: Websites of Online Resource

Websites	Link
Google	https://www.google.com/
MITRE ALTAS	https://atlas.mitre.org/
GitHub Blog	https://github.blog/
JFrog	https://jfrog.com/blog
Sonatype	https://www.sonatype.com/blog
Synk	https://snyk.io/blog/
Tencent Security	https://security.tencent.com/index.php/blog
Hacker News	https://thehackernews.com/
Checkmarx	https://checkmarx.com/blog/
Check Point	https://blog.checkpoint.com/
BleepingComputer	https://www.bleepingcomputer.com/news/security
The Register	https://www.theregister.com/security
Security Intelligence	https://securityintelligence.com/news/
heise	https://securityintelligence.com/news/
SC Magazine	https://www.scmagazine.com/

of cybersecurity, professionalism, timely updates, and authoritative information. On the one hand, we leverage the same keywords used in the academic literature collection on the websites containing querying interfaces. *e.g.,* Google. On the other hand, we manually inspect the websites to collect the relevant news or events.

3.2 Literature Analysis and Taxonomy Construction

We manually inspected and categorized various elements of large language model risks and mitigations from the literature. First, we used open card sorting in our analysis. We began by thoroughly reviewing the literature set, and identified and highlighted terms, concepts, techniques, and strategies related to the large language model risks and mitigations.

We performed hierarchical tree structure construction, where we connected the identified categories into a structured, hierarchical framework that reflects the relationships between different risks and mitigation elements. We organize these categories into tree structures where the root represents the risks, stakeholders, and mitigations. To ensure accuracy and comprehensiveness, we refine our summarized structures according to the following criteria with multiple iterations.

- **Mutual Exclusivity.** Assess whether each node is distinct from the others.
- **Accuracy.** Verify the precision of node descriptions by consulting attack case studies and expert feedback, ensuring alignment with real-world attack patterns.
- **Hierarchy.** Evaluate whether the tree's structure is logical, ensuring broader attack categories at higher levels and specific methods at lower levels.
- **Comprehensibility.** Test the clarity of node descriptions through expert panels or user feedback, ensuring that they are easy to understand.
- **Relevance.** Ensure that each node reflects the latest risks and mitigations by reviewing recent research and industry standards (*e.g.,* CVE, CWE, MITRE ATT&CK).
- **Coverage.** Check whether the node includes all known attack techniques for its type, comparing against existing security frameworks and databases.

Each node in the structures is assessed by the four professionals independently. The structures are refined and re-evaluated iteratively until all professionals reach a final agreement.

4 Risks of Large Language Model Supply Chain

In this section, we summarize the risks in the LLM supply chain through illustrative risk scenarios. The scenarios are presented as *risky stakeholders* performing *risky actions* on specific *supply chain components*, which in turn lead to distinct *risk types*. We first discuss the *risky stakeholders*. Then, we study the *risk types*. Next, we propose the taxonomy of *risky actions* in the large language model supply chain. The *risky stakeholders* are detailed in the Section 2.2.1 and the *supply chain components* are covered in the Section 2.2. The *risky actions* are illustrated in Section 4.2 and the *risk types* are illustrated in Section 4.3. We present the risk scenarios regarding *Risky Contributors*, *Risky Consumers*, *Risky Administrators* and *Risky Users* in Figure 5, 6, 8 and 7, respectively.

4.1 Risky Stakeholders

We discuss the risky stakeholders in terms of risky contributors, risky consumers, risky administrators and risky users.

- *Risky Contributors.*
  - *Camouflage Contributors.* These contributors attempt to blend in with legitimate contributors, possibly hiding their true intentions.
  - *Attack Contributors.* This group actively seeks to compromise the system by injecting malicious code, data, or models into the supply chain.
  - *Being Benign Contributors.* Contributors in this category introduce vulnerabilities unintentionally, either through poor coding practices, lack of maintenance, or other factors that increase system risk.
- *Risky Consumers.*
  - *Being Middleman.* Entities that act as intermediaries, facilitating the exchange or integration of supply chain components between contributors and other consumers.
  - *Being Artifact Consumer.* These stakeholders use artifact (*e.g.,* data) for training, analytics, or other purposes.
  - *Being Platform Consumer.* Entities that rely on platforms built with supply chain components, such as development or deployment environments.
- *Risky Administrators.*
  - *Attack Administrators.* Attackers can hack into an administrator's account, causing severe consequences given the elevated privileges and control administrators hold over systems.
  - *Being benign administrators.* Administrators who despite having good intentions, inadvertently introduce risks into the system through misconfigurations or other unintentional actions.
- *Risky Users.* Users can be risky if their purposes are to exploit LLM application vulnerabilities, steal models, conduct jailbreaking or inference attacks, perform excessive access, or pollute the data using user feedbacks.

4.2 Risk Types

The LLM supply chain risks are classified into four main types. *i.e.,* Security Risks, Privacy Risks, Delivery Risks, and Legal Risks.
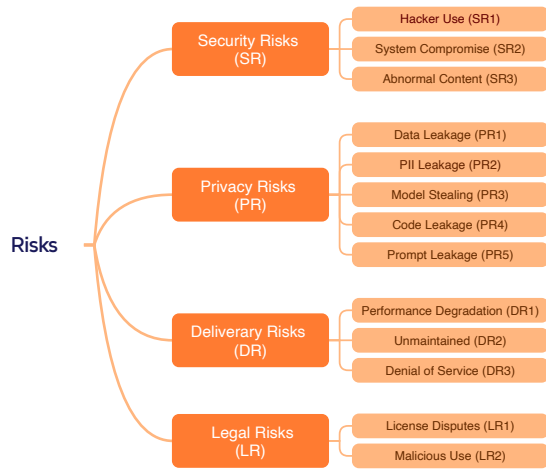
Figure 4: Risk Types

Figure 4 provides a comprehensive overview of the LLM supply chain risks across security, privacy, operational, and legal domains.

- **Security Risks (SR)** include potential threats such as hacker use (SR1), system compromise (SR2), and the emergence of abnormal content (SR3) that could disrupt systems or compromise integrity.
- **Privacy Risks (PR)** focus on data vulnerabilities, such as data leakage (PR1), personally identifiable information (PII) leakage (PR2), model stealing (PR3), code leakage (PR4), and prompt leakage (PR5), all of which can result in the unauthorized access or misuse of sensitive information.
- **Delivery Risks (DR)** highlight issues related to the performance and maintenance of services, including performance degradation (DR1), system unmaintenance (DR2), and denial of service (DoS) (DR3) attacks that hinder service availability.
- **Legal Risks (LR)** emphasize complications arising from copyright disputes, license disputes (LR1), and malicious use (LR2) of systems, which can lead to legal and compliance challenges.

4.3 Taxonomy of Risky Actions

We present the taxonomy of LLM supply chain risks with regard to the six composing elements in the LLM supply chain.

4.3.1 Risks on Code. The code, including packages, frameworks, and plugins, have become essential at every stage of LLM application development. However, using these libraries comes with potential risks.

Implant Malicious Code. Attackers can develop malicious code and attempt to implant it through various methods, finally integrating it into the LLM supply chain. For instance, Zhao et al. [127] revealed 9 malicious dataset loading scripts on Hugging Face, and perform root cause analysis of vulnerable formats. It can cause the risk of malicious use (LR2).
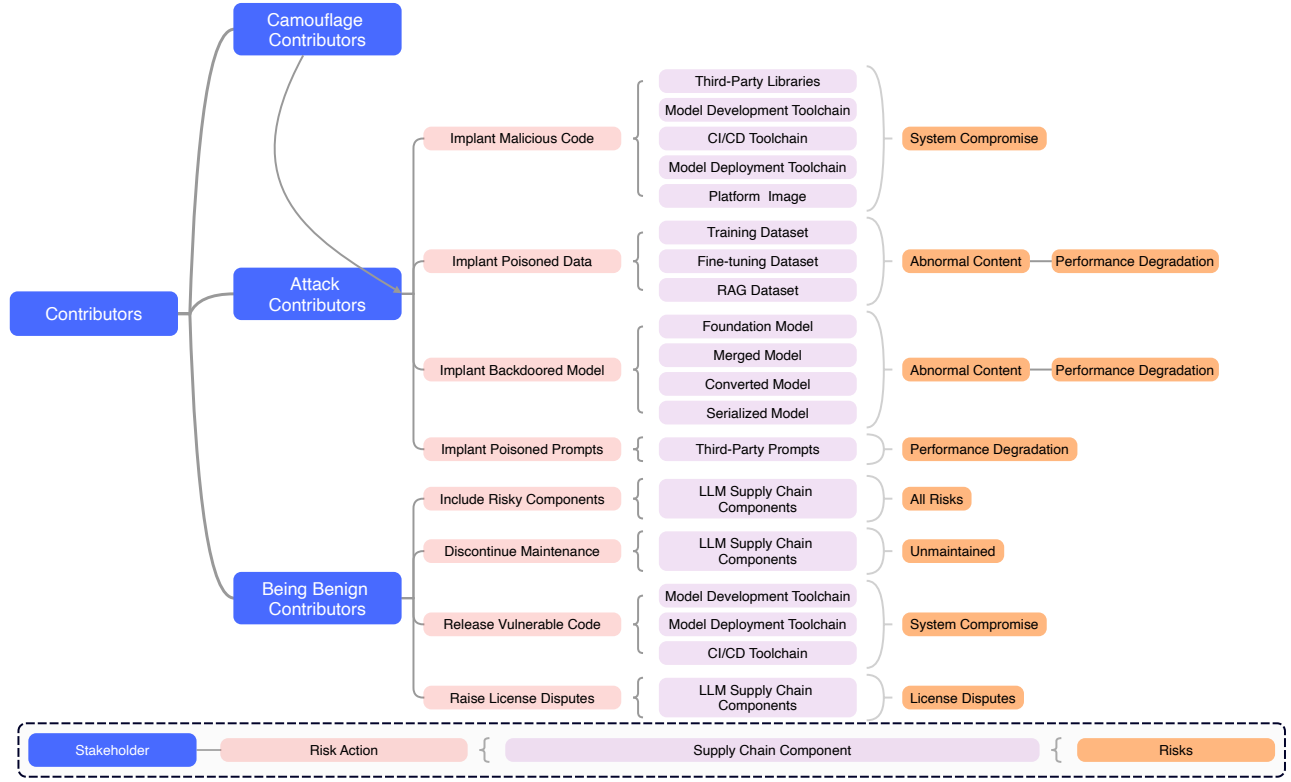
- *Dependency Confusion.* A dependency confusion attack is a type of supply chain attack where an attacker publishes a malicious package to a public package repository with the same name as a popular private package. The package manager may unknowingly download the malicious package from the public repository instead of the intended private repository [85].

- *Typosquatting.* Using typosquatting or combosquatting, attackers upload malicious packages with names that are nearly identical to popular legitimate packages. A minor typographical error during package installation can result in the inclusion of malicious code in the application. For example, PyPI has suffered from numerous malicious typosquatting packages. These malicious packages can lead to the leakage of Personally Identifiable Information (PII) and compromise the integrity and security of systems [12]. Rokon et al. [88] identified over 7,500 instances of malware hosted on GitHub. Attackers may also employ more covert methods, such as spreading malware through forks [9].

- *Compromising Legitimate Libraries.* Attackers may target legitimate libraries already integrated into the LLM supply chain, leveraging their established presence and substantial user base. Attackers can conceal their malicious code within pull requests, subtly injecting harmful elements into the codebase. In addition, attackers may take control of the code from a previous maintainer and release new malicious versions [45].

- *Sharing Cloud Environments.* Attackers can inject malicious code into cloud environments and make it publicly available to all cloud users. The rise in popularity of containers (*e.g.,* Docker) has accelerated the trend of deploying applications in cloud environments. However, it in turn has become a hugely popular attack vector recently. The research team discovered that over a thousand Docker container images were found hiding malicious content [17]. Tomar et al. [106] proposed a taxonomy of potential attacks at the container layer, revealing different types of threats that can arise within containerized environments.

4.3.2 Risks on Dataset. The datasets play a fundamental role, providing the extensive data required for model training, model refining and prompt generation.

Implant Poisoned Data. Attackers can create poisoned data and implant it through various methods, with the aim of having it used during critical stages of building the LLM supply chain. It can cause the risk of data leakage (PR1) and malicious use (LR2).

- *Data Poisoning.* Attackers can manipulate the data for malicious purposes, and then upload it to public platforms (*e.g.,* Kaggle) [64]. Given that training large language models requires vast amounts of data, it is highly likely that these poisoned data sets will be used by LLM model developers. If the poisoned data is unintentionally leveraged into the LLM supply chain, it would result in the model performance degradation, particularly in its ability to generate the outputs accurately and efficiently [92, 95, 110]. Alexander et al. [109] demonstrated that with only 100 poisoning examples, LLM produces consistent negative results or vulnerability output across hundreds of tasks.

- *RAG Poisoning.* Retrieval-augmented generation (RAG) is widely incorporated in the LLM supply chain to provide the LLM with sufficient contextual knowledge. Similarly, the knowledge database can also be poisoned, where it can be manipulated the guide the LLM to generate specific, attacker-chosen responses to particular questions. Zou et al. [131] proposed PoisonedRAG, which specifically focuses on injecting poisoned texts into knowledge databases to steer language models to respond with predefined

Figure 5: Risky Actions, Supply Chain Components and Risk Types Introduced by Risky Contributors

answers to questions selected by the attacker. Wang et al. introduce the concept of Poisoned-LangChain [113], a novel indirect attack that exploits a poisoned external knowledge base.

4.3.3 Risks on Models. The open-source models provide accessibility, flexibility, and transparency, allowing developers to modify, customize, and integrate them into various projects.

Implant Model Backdoors. Attackers can manipulate publicly accessible models to alter their behavior for malicious purposes, ultimately affecting downstream users who rely on the compromised models. LLM may be composed of multiple components, including vocabularies, tokenizers, embedding layers, and auxiliary models (*e.g.,* CLIP-ViT), etc [13, 62, 86]. It can cause hacker use (SR1), system compromise (SR2), emergence of abnormal content (SR3), privacy risk (PR), data leakage (PR1), personally identifiable information (PII) leakage (PR2), model stealing (PR3), code leakage (PR4), and prompt leakage (PR5).

- *Component Dependency Attack.* Attackers can manipulate the contents or structures of the composing models of LLMs, thereby conducting component dependency attacks. Huang et al.[44] firstly explore how manipulating the embedding dictionary using carefully designed rules could cause the model to produce specific outputs. Additionally, Cui et al.[20] and Shayegani et al. [93] demonstrated that LLMs are not robust against attacks targeting the visual models used exclusively in multi-modal LLMs.
- *Backdoor attack.* Backdoor attack embeds hidden backdoors in the model during the training process, allowing the compromised model to perform normally on benign samples but altered on the

hidden backdoor input [16, 60, 80]. Yao et al. and Cai et al. [8, 120] studied the vulnerability of prompt learning algorithms commonly used in LLM to backdoor attacks.

Inference Attack. Inference attacks are types of privacy attack targeting data, where the responses of LLM models may inadvertently leak sensitive information, leading to unauthorized access, intellectual property theft, and privacy breaches. They include attribute inference attacks and membership inference attacks. Attribute inference attacks involve deducing sensitive information (*e.g.,* race, gender, and sexual orientation) from the behavior or responses of LLM models, even if such information is not explicitly included in the training data. Robin et al. [99] and Pan et al. [79] conducted systematic studies on various advanced large language models, which have shown that LLMs can accurately infer sensitive data such as identity, genome, healthcare, and location information without any prior knowledge. Membership inference attacks aim to predict whether a data sample was included in the training data of LLM, undermining the trust between data providers and users. Mattern et al. [65] explored the significant threat of membership inference in the real-world using neighborhood comparison. It can cause data leakage (PR1), personally identifiable information (PII) leakage (PR2), model stealing (PR3), code leakage (PR4), and prompt leakage (PR5).

Model Stealing. Model stealing refers to attacks where an adversary attempts to extract sensitive information about a machine learning model (*e.g.,* model gradients, training data, and model

parameters), thereby compromising the model's privacy. It encompasses various model privacies, including gradient leakage [39], training data extraction [119, 122, 126], and model information extraction [10]. It not only infringes the intellectual property rights of the model owner but also leads to security risks. For example, adversaries can create shadow models via model stealing, which are then used to generate adversarial examples, reverse-engineer sensitive data used in model training [79], or bypass any access restrictions or usage limitations in the original model [94]. It can cause the privacy risk of model stealing (PR3).

Regarding LLMs, several model-stealing attacks have been reported to be effective [82, 119, 126]. Li et al. and Yu et al. [11, 122] conducted experiments to investigate the effectiveness of training data extraction on LLMs. Truong et al. [107] proposed techniques from the domain of data-free knowledge transfer to perform model extraction, successfully replicating the capabilities of the original model. Carlini et al. [10] demonstrated precise and significant information extraction from black-box generative language models by recovering the embedding projection layer of a transformer model, given typical API access.

4.3.4 Risks on Prompts. Since LLMs are typically built from general knowledge, prompts are an effective way to incorporate domain-specific insights, directly enhancing the quality, relevance, and accuracy of the model's output. The shared prompts allow a wider community to contribute and share specialized knowledge. However, shared prompts present several risks. In the context of shared prompts, those jailbreak prompts can be maliciously leveraged and pose challenges to the security mechanism of existing LLMs.

Implant Poisoned Prompts. Implanting poisoned prompts deliberately inserting harmful or misleading prompts into publicly accessible repositories or datasets. These poisoned prompts can be designed to manipulate the behavior of large language models (LLMs) by influencing their output in undesirable ways. Since shared prompts are widely shared and reused, such poisoned prompts can easily propagate through the supply chain, leading to inaccurate, biased, or even harmful model responses for downstream users who incorporate them into their applications. It can cause the performance degradation (DR1) of dilivery risks.

- *Prompt injection.* Prompt injection is a type of attack that exploits malicious prompt words with trusted prompts against LLM applications. By exploiting the incapability of distinguishing "good" and "bad" prompts, prompt injection can cause malicious inputs to affect the model's behavior and output, leading to information leakage or other security vulnerabilities. Stumpp et al.[37] demonstrated the potential of prompt injection to indirectly execute arbitrary code within a Python interpreter in LLM applications, thereby gaining control over the host server (*e.g.,* accessing environment variables, arbitrary file contents, executing DoS attacks, etc.). Chen et al.[14] and Shayegani et al. [93] further explored the threats of prompt injection using multi-modal inputs.

- *Jailbreaking.* Jailbreaking attempts to subvert safety filters built into the LLMs themselves [115]. Recent research [21, 114, 130] has used a variety of methods to make the LLMs LLMs responsive to restricted or insecure content (*e.g.,* content that contains pornography, biased content, or assists the user in committing criminal acts). Wei et al. [114] investigates the reasons for the
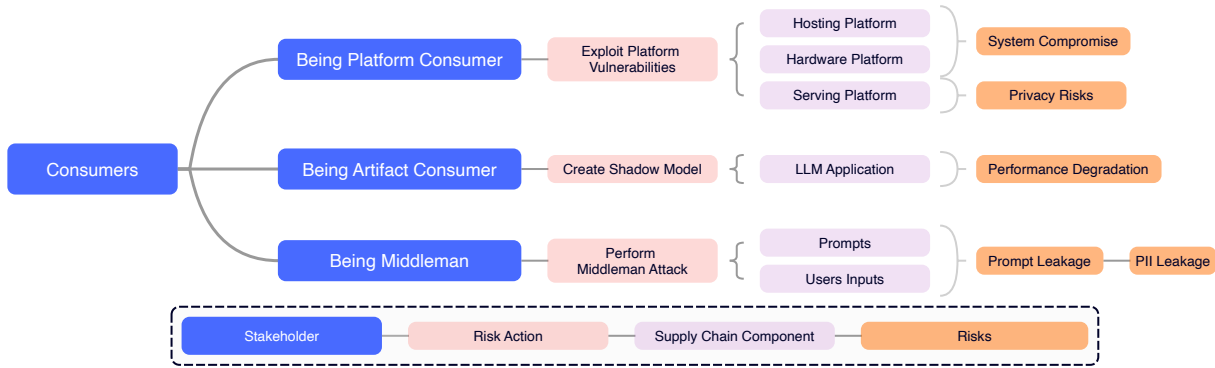
success of such attacks and how they occur, emphasizing the need for safety-capability parity. To assess the potential harm caused by jailbreaking, Shen et al. [94] devised the JailbreakHub framework including 1,405 jailbreak prompts.

4.3.5 Risks on LLM Applications. The LLM application performs various tasks using the LLM involving natural language processing, task-based dialogue, database search, etc. It can be a web application, an agent or a chatbot. The potential risks may be exposed in multiple ways.

Exploit LLM Application Vulnerabilities. The LLM Plugin Store (*e.g.,* GPT store [76]) is a platform where users can discover, install, and manage plugins designed for enhancing the functionality of large language models. However, attackers can publish malicious plugins, guiding the model to output incorrect content through prompt injection [116]. Moreover, some third-Party plugins could lead to account takeovers and data leakage [51, 52, 73]. Exploit lLM application vulnerabilities can potentially cause the hacker use (SR1), data leakage (PR1) and performance degradation (DR1).

Excessive Access. Users may overload the LLM application with excessive requests, potentially causing it to malfunction or become unresponsive. A Denial of Service (DoS) attack is a type of cyberattack that seeks to deplete computational resources, resulting in latency or rendering resources unavailable. Given that large language models (LLMs) demand substantial computational power, attackers can deliberately craft prompts to diminish the availability of these models [22]. Shumailov et al. [96] demonstrated the feasibility of performing DoS attacks within the domain of LLMs. These attacks significantly increased the energy consumption and latency of the model (boosted to 10 to 200 times), raising researchers' concerns about performance security in critical decision-making scenarios. Stumpp et al. [100] et al. used prompt injection to control the execution of dead-end code by the code executor of an LLM application, resulting in an unresponsive program host server. Excessive access can potentially cause hacker use (SR1), system compromise (SR2), emergence of abnormal content (SR3), data leakage (PR1), personally identifiable information (PII) leakage (PR2), model stealing (PR3), code leakage (PR4), and prompt leakage (PR5), and performance degradation (DR1).

Feedback Pollution. Existing LLM systems are often tweaked based on user feedback to improve their performance [38]. However, attackers may disrupt the performance of an LLM system by injecting a large amount of irrelevant data, leading to a surge in false feedback. Consequently, analysts at the victim organization may find themselves spending excessive time reviewing and rectifying erroneous inferences [68]. More seriously, LLM model developers may use this erroneous feedback during the instruction tuning phase to conduct reinforcement learning based on human feedback. This situation can not only lead to a decline in model performance but also cause the model to exhibit unpredictable behavior when facing real-world data [5, 109]. Sun et al. [101] demonstrate two proof-of-concept examples, stealthily manipulating a neural code generation system to generate code with vulnerabilities, attack payload, and malicious and spam messages. Feedback pollution can potentially cause the hacker use (SR1), system compromise (SR2), emergence of abnormal content (SR3), privacy risk (PR).

Figure 6: Risky Actions, Supply Chain Components and Risk Types Introduced by Risky Consumers

Perform Middleman Attack. A Man-in-the-Middle (MitM) attack involves an attacker intercepting communication between two parties without their knowledge. In the context of using the LLM Application or any chatbot platform, a MitM attack could occur if an attacker gains access to the communication channel, allowing them to alter or steal sensitive information being shared between a user and the AI system. Kang et al. [49] demonstrated that attackers may intercept data transmitted during a user's interaction with LLM-based applications, leading to unauthorized access to sensitive information. Si et al. [97] proposed that attackers can set up a fake API that mimics a legitimate one while offering a lower price to lure users. It not only allows the attacker to modify or steal sensitive data but also to exploit economic gains from the middle. The middleman attack can potentially cause data leakage (PR1), personally identifiable information (PII) leakage (PR2), model stealing (PR3), code leakage (PR4), and prompt leakage (PR5).

4.3.6 Risks on Platforms. Many platforms play important roles in the LLM supply chain, supporting various stages such as source code management, data hosting, model training, monitoring, and deployment. However, these platforms can introduce inherent risks.

Camouflage Contributors. Attackers can create fake accounts that imitate existing platform accounts (e.g., official accounts) to publish malicious data, code, models, and more. Users may be misled by these repositories, inadvertently using malicious tools to develop or deploy large models. It can cause the risk of malicious use (SR1).

Malicious Contributors. Attackers can obtain and exploit credentials of existing accounts or API tokens through various methods [71, 72]. An LLM red team organization [105] has demonstrated in practice that once in possession of these credentials, attackers can gain unauthorized access to a range of resources and services associated with LLMs, potentially compromising the integrity, confidentiality, and availability of the targeted systems and data. Attackers can also use social engineering tactics to become maintainers of legitimate projects. Once they have obtained maintainer status, they can introduce malicious code or dependencies, which may go unnoticed by other contributors or users due to the perceived legitimacy of the new maintainer. For instance, in the XZ attack, the attacker gained maintainer access to the XZ project, embedding a backdoor in a critical dependency [4]. It can cause the risk of malicious use (SR1).

Exploit Platform Vulnerabilities. Exploiting platform vulnerabilities refers to attackers identifying and taking advantage of weaknesses in the underlying infrastructure of an LLM or related application. These vulnerabilities can exist in the platform's software, hardware, or configuration, allowing attackers to perform unauthorized actions such as gaining access to sensitive data, altering model behavior, or disrupting services. It can cause the risk of hacker use (SR1), data leakage (PR1) and model stealing (PR3).

- *CI/CD Flaws.* Attackers can leverage flaws in AI cloud services to carry out attacks. Researchers from Wiz identified vulnerabilities in SAP's AI Core service, which could have allowed attackers to access sensitive data of other tenants within SAP's cloud infrastructure [90]. By exploiting these flaws, attackers can retrieve AWS tokens, access AI training data, and potentially cause data theft, service disruption, and even supply chain attacks within LLM applications.

- *Privacy Leakage.* Attackers can exploit secrets found in public artifacts generated by automated CI/CD pipelines. Recent research revealed that artifacts produced by GitHub Actions workflows may contain sensitive tokens, such as GITHUB_TOKEN and ACTIONS_RUNTIME_TOKEN, which attackers could use to inject malicious code [31]. When users download and utilize these compromised artifacts, it can result in a supply chain attack.

4.3.7 General Risks to LLM Supply Chain Component. Apart from risks that are specific to the six composing elements, we elaborate on general risks to LLM supply chain components.

Introduce Risky Components. A benign contributor may not realize that the components they introduce have hidden vulnerabilities or security flaws. These risky components can be exploited by malicious actors once deployed. For example, an LLM application might rely on a library that appears secure but contains a hidden vulnerability. When the benign contributor integrates this library, they inadvertently expose the LLM and its users to potential exploitation. Attackers can leverage these risky components for data theft, model manipulation, or unauthorized access. It can cause the risk of hacker use (SR1), data leakage (PR1) and model stealing (PR3).

Discontinue Maintenance. When a contributor stops maintaining a component, updates for security patches, bug fixes, and compatibility improvements are halted. It can lead to vulnerabilities if the abandoned component contains outdated dependencies that are
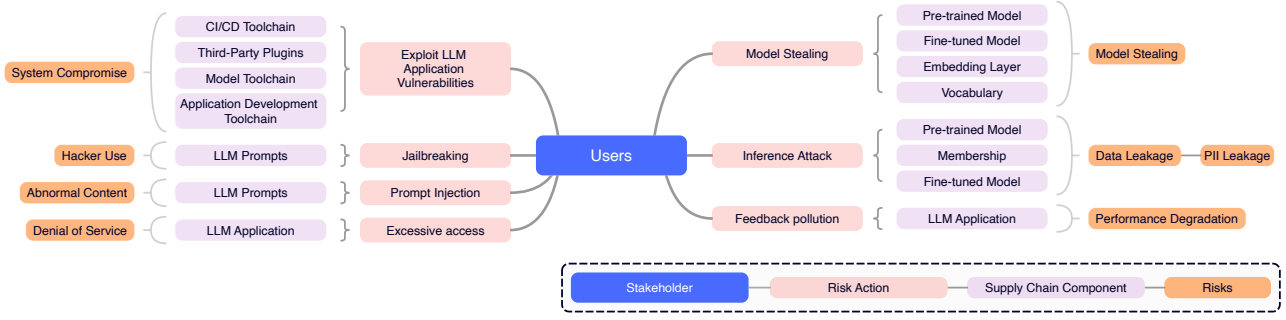
Figure 7: Risky Actions, Supply Chain Components and Risk Types Introduced by Risky Users
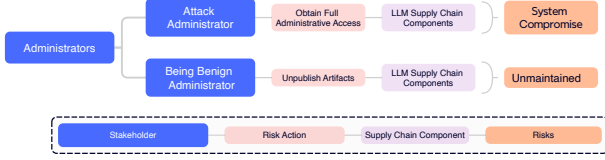

Figure 8: Risky Actions, Supply Chain Components and Risk Types Introduced by Risky Administrators

prone to exploitation. Moreover, the discontinuation can disrupt further development by creating compatibility issues, reducing model performance, or complicating integration with newer systems. Furthermore, it poses significant risks because malicious actors could take advantage of this by forking the abandoned project and inserting malicious code. It can cause the risk of unmaintained (DR2).

Release Vulnerable Code. The benign contributors may submit code that contains security flaws or exploits without malicious intent. Such vulnerabilities could range from weak encryption practices, insecure data handling, or exploitable logic errors. While the contributor's intent is not harmful, the inadvertent introduction of security issues can still have serious consequences, potentially leading to supply chain attacks. It can cause hacker use (SR1), system compromise (SR2), and the emergence of abnormal content (SR3).

Raise License Disputes. While open-source licenses generally grant permission to use, modify, and distribute the code, the owners retain the copyright. It introduces potential risks for users who download and use the code, model or data if contributors claim copyrights on their code. They may enforce restrictions or take legal action if users violate the terms of the license. In 2020, CoKinetic Systems sued Panasonic for allegedly violating the GPLv2 license, with the lawsuit seeking financial compensation [69]. It can cause the risk of license disputes (LR1).

5 Mitigation of Risks in Large Language Model Supply Chain

We elaborate on the mitigation of risks in the LLM supply chain with regard to the seven aspects.

5.1 Mitigation to Risks from Third-party Libraries

Malware Detection. Malware detection identifies viruses, worms, trojans, and backdoors from third-party libraries in both binaries and source code. Key techniques include signature comparison,

static analysis, and dynamic analysis. Signature-based detection tools, like VirusTotal [108], are effective for identifying binary malware. To detect malware within package registries, static and dynamic analyses are widely applied. Recent research has advanced these approaches by modeling the behavior of malicious packages using static characteristics alone [43, 124], as well as integrating both static and dynamic analyses to enhance detection accuracy [23, 40]. This measure can mitigate the risk SR2.

Software Bill-of-Materials Analysis. A Software Bill of Materials (SBOM) is a list of inventory components included in imported third-party libraries. SBOM analysis provides visibility into all third-party libraries and their versions, helping users avoid unvetted or potentially risky libraries. Several commercial tools offer SBOM analysis, including Sonatype Management [98], BlackDuck [6], and Microsoft's SBOM tool [66]. By utilizing SBOM analysis, users can enhance security and improve overall software quality throughout the development lifecycle. This measure can mitigate the risk SR2 and SR3.

5.2 Mitigation to Risks from Dataset

Data-Deduplication. In the face of privacy risks, we can leverage data deduplication on the dataset. Jagielski et al. discovered that model privacy attacks are more effective on the training dataset that appears multiple times [47]. Removing duplicate text from the training data [1] helps protect against model memorization of sensitive information, making privacy attacks less effective. This measure can mitigate the risk PR2.

Data Encryption. Data encryption protects data confidentiality by converting it into an unreadable format, consisting of three types: symmetric encryption [41], asymmetric encryption [42, 61], and hybrid encryption [2]. With effective key management mechanisms, the data cannot be deciphered, significantly reducing the risk of data privacy breaches. This measure can mitigate the risk PR1 an PR2.

Data Sanitization. Data sanitization cleans input data to defend against data poisoning and backdoor attacks by applying techniques like data denoising, filtering, and smoothing, which effectively remove adversarial noise while retaining legitimate information [118]. Additionally, these methods help prevent data leakage by ensuring that shared or processed data cannot be traced back to individuals. This measure can mitigate the risk SR3 and PR1.

Data Watermark. Data watermark protects data privacy by embedding specific watermark identifiers [102, 104] into the training dataset. During the training process, the model is trained using

watermarked data, ensuring that it generates output text containing the corresponding watermark when given specific inputs, thus achieving the goal of identifying and tracing the data source. This measure can mitigate the risk PR1.

Differential Privacy. Differential privacy [3] protects the private information of individual records in a dataset [24, 25]. Specifically, a differential privacy mechanism adds noise to each training data point. It ensures that even if a particular data point is replaced or removed, the model's output will not be significantly affected, which effectively prevents attackers from inferring private information in the training data by analyzing the model's outputs. This measure can mitigate the risk PR2.

5.3 Mitigation to Model Risks

Model Signature. Model signature helps to track and manage the origin of models. Jiang et al. [48] described useful attributes for representing the model signature, such as provenance, reproducibility, and portability. Recording and verifying key attributes, provides a reliable reference point, allowing researchers or engineers to trace and verify the model's training process, thereby determining whether the model has been maliciously modified or compromised. This measure can mitigate the risk PR3.

Model Watermark. Model watermark embeds traceable information into the text generated by LLMs to verify the model's source and authenticity without degrading output quality. Techniques such as logits modification [50], entropy-based embedding [57], multibit payload [111], and token sampling [53] enable efficient watermarking by embedding verifiable identifiers that protect against model-stealing, safeguarding intellectual property and ensuring model integrity. This measure can mitigate the risk PR3.

Model Obfuscation. Model obfuscation protects the security and privacy of the LLMs. It modifies the model's structure or parameters to prevent attackers from being reverse-engineered or maliciously exploited. Zhou et al. [129] proposed an active defense solution for model theft via automated weight obfuscation. This measure can mitigate the risk PR3.

Model Alignment. Model alignment ensures that a model's outputs and behaviors align with human values and expectations. Reinforcement Learning from Human Feedback [77] incorporates human feedback into the model's training process, guiding the model to learn which behaviors are appropriate or not. This measure can mitigate the risk SR3.

Adversarial Training. Adversarial training is a machine learning technique designed to enhance model robustness by exposing the model to adversarial examples [117] during training. By learning to recognize and handle these malicious inputs, the model becomes more resilient to attacks and unpredictable behavior. This measure can mitigate the risk SR3.

Federated Learning. Federated Learning is a decentralized approach where multiple clients collaboratively train a shared model without transferring their individual data to a central server. It prevents the aggregation of malicious updates by identifying and excluding anomalies in the distribution using unsupervised or supervised machine learning techniques [59, 125]. By leveraging three-stage defenses (pre-aggregation defense (Pre-AD), in-aggregation defense (In-AD), and post-aggregation defense (Post-AD)), federated learning ensures that the aggregated model remains secure and resistant to backdoor attacks or malicious data manipulation, while still benefiting from decentralized data training. This measure can mitigate the risk SR3.

5.4 Mitigation to Prompt Risks

Prompt Sanitization. Prompt sanitization filters, formats, or regularized prompts are provided to LLMs to ensure that the model does not misinterpret their original intent, thereby preventing abnormal content or malicious behavior. [87, 103] have been shown to effectively prevent jailbreak in LLMs. Suo et al. [103] applied to sign to sensitive elements (*e.g.,*'delete file') in the user's prompt. Robey et al. [87] randomly perturb characters in a given input prompt (e.g., inserting random characters). Chen et al .[14] separates prompts and user data into two independent channels to prevent prompt injection. This measure can mitigate the risk SR3.

Robust Tuning. Robust tuning enhances the defense of LLMs against prompt risks by incorporating specific techniques during tuning. For example, Yue et al. [123] fine-tuning a generative language model with differential privacy to generate privacy-preserving synthetic text. Ozdayi et al. [78] leverage prompt-tuning to control the extraction rates of memorized content, avoiding attack modifying LLM weights. This measure can mitigate the risk SR3.

Model Structure Optimization. Model structure optimization aims to enhance the robustness of LLMs' inference by modifying their structure to defend against malicious prompts. Chen et al [15]. Employ homomorphic encryption to enable privacy-preserving inference for Transformer-based models. Li et al. [58] achieve efficient and private Transformer inference using Secure Multi-Party Computation (MPC). This measure can mitigate the risk PR1.

5.5 Mitigation on LLM Applications Risks

Secure Auditing and Permission Management. Secure auditing and permission management are crucial for mitigating risks associated with LLM plugins. This approach includes enforcing strict vetting processes to ensure that only trusted plugins are integrated into the system, implementing anomaly detection systems to identify unusual behaviors indicative of malicious activity, and conducting regular security audits to assess plugin security and compliance. Additionally, employing role-based access control (RBAC) limits permissions, ensuring that users and plugins have only the necessary access to perform their functions, thereby reducing the attack surface. By maintaining a comprehensive logging system, organizations can track interactions and quickly respond to potential security incidents, fostering a proactive security posture. This measure can mitigate the risks SR1, SR3 and LR2.

Authorized and Restricted Access. Authorized and restricted access is a key defense mechanism against several risks in LLM applications. For feedback pollution, restricting access to authorized users ensures that only credible feedback is collected for model training. For Middleman (MitM) attacks, by enforcing strong access controls, such as mutual authentication and encryption, only authorized clients can communicate with the LLM. For excessive access, limiting access to verified users via rate-limiting mechanisms

mitigates the risk of excessive API requests that can overload the system, leading to Denial of Service (DoS) attacks. By ensuring only authorized users can access the LLM, and controlling the frequency and scope of interactions, systems can prevent abuse and maintain availability. This measure can mitigate the risks DR1, DR2 and DR3.

5.6 Mitigation to Platform Risks

Anomaly Monitoring. Platforms can apply authorized and restricted access for untrusted accounts. *e.g.,* multi-factor authorization [84]. Apart from that, they can adopt anomaly monitoring by observing developing activities to identify any suspicious behavior, such as unauthorized access attempts, changes to account settings, or unexpected pull requests. For example, Danielle et al. [36] leveraged commit messages to detect anomalies and malicious commits. This measure can mitigate the risk SR2.

Mitigating Platform Vulnerabilities. Platforms should also ensure that their CI/CD systems are following best security practices, such as least privilege, secure credential storage, and network segmentation, keeping all CI/CD tools and dependencies up to date with the latest security patches to protect against known vulnerabilities [90], and regular security assessments and vulnerability scans of the CI/CD environments to promptly identify security flaws. This measure can mitigate the risk SR2.

Session Isolation and Privacy Protection. Using sandboxing in platforms isolates user sessions within separate environments, preventing cross-session attacks. By restricting access to shared resources and data, sandboxing ensures that malicious activities in one session cannot affect others. This containment helps protect sensitive information and enhances overall security by limiting the impact of potential breaches. Besides, platforms can utilize secrets management tools, log scanning, and sanitization to ensure that sensitive information like tokens and API keys are not exposed in repositories or logs [31]. This measure can mitigate the risks SR2, PR1, PR2, PR3 and PR4.

5.7 Mitigation to General LLM Supply Chain Component Risks

Enhancing Security and Legal Risk Awareness. Developers and organizations should be vigilant when incorporating components into their software systems. On the one hand, they should actively monitor for known vulnerabilities in third-party libraries and frameworks, thus reducing their exposure to potential threats. On the other hand, they should familiarize themselves with licensing agreements, intellectual property rights, and the potential liabilities tied to both open-source and proprietary components. This measure can mitigate the risks SR2 and LR1.

Training for Secure Coding. Adhering to secure coding practices is fundamental to minimizing software vulnerabilities. This includes following best practices for input validation, implementing robust error handling, and adhering to established secure coding guidelines. By prioritizing these practices, developers can substantially lower the likelihood of introducing security flaws into their applications, thereby safeguarding user data and assets. This measure can mitigate the risk SR2.

Preparing Alternative Components. To mitigate the risks associated with reliance on components that may become unsupported or discontinued, it is important to seek alternative solutions. Moreover, advanced replacement techniques are also needed to help migrate the original component into a new one with the least effort and cost. By diversifying their dependencies and preparing for potential discontinuations, the stability and security of the LLM-driven application can be enhanced. This measure can mitigate the risks SR2, DR2 and DR3.

6 Discussion

We discuss future challenges and opportunities towards mitigating the risks in the LLM supply chain.

6.1 Future Challenges

Comprehensive and Precise Decomposition of the LLM Supply Chain. Decomposing the LLM supply chain is essential in understanding and mitigating risks in the LLM-related ecosystem, ensuring compliance, and enhancing transparency across the LLM lifecycle. First, *mapping an entire LLM supply chain is a complex task.* As the scope and applications of LLMs expand, defining clear boundaries for the LLM supply chain becomes nearly impossible. Nevertheless, identifying a clear scope of relevant components is crucial to enable a comprehensive analysis of the LLM supply chain, including its core artifacts, stakeholders, and supply relationships. This clarity allows each stakeholder to understand their role and responsibilities within the supply chain effectively. Second, *the unique dependencies of the LLM supply chain adds further complexity.* Unlike traditional software supply chains, the LLM supply chain includes special dependencies—such as models embedding external knowledge (e.g., data or code). The models do not directly use external resources. Instead, they encode knowledge into tensors and weights, obscuring the origins and composition of these elements. Third, *LLM artifacts (e.g.,* models, datasets, and prompts) *differ fundamentally from software artifact, making it difficult to directly apply standard version management.* This limitation complicates efforts to decompose the LLM supply chain effectively and hinders accurate tracking and management.

Compliance with Government Laws and Guidelines. The growth of new LLM applications is expected to accelerate with the increasing availability of commercial and open-source LLMs, leading to their adoption across a wide range of industries from different countries. Since industries across countries may be subject to varying regulations and requirements, ensuring compliance with government laws and guidelines is essential. First, as LLMs often rely on data sourced from various online platforms, *verifying the legality of training datasets to see if it is intellectual property (IP) free is essential.* Given that the conclusion of dataset composition may be indirect, it is challenging to establish a common sense for copyright infringement or IP violations solely based on the model's generated output or trained weights. Furthermore, the providers may deliberately wipe traces of the IP for their benefit. Therefore, it is also crucial to fight against those actions. Second, *the vast data required to train LLMs includes sensitive information that may pose privacy risks.* Ensuring compliance with data privacy laws like GDPR, alongside securing data through encryption and

anonymization is also a critical issue. Third, *LLMs are prone to inheriting biases present in their training data*, which can persist in the downstream application, affecting their performance and trustworthiness. Developing robust strategies for identifying and mitigating biases is crucial for ethical AI.

Vigilant Prevention of Adversarial Attacks. On one hand, with reliance on third-party data sources, libraries, proprietary frameworks, and cloud providers, *there is a risk of supply chain vulnerabilities and limited control over model updates*. This creates security concerns as well as challenges in maintaining compatibility and transparency. On the other hand, *LLMs are increasingly targeted by adversarial attacks*, such as data poisoning or model inversion, which can compromise data integrity or privacy. Developing robust defenses will be necessary to ensure secure usage.

6.2 Opportunities

Collaborative Open-Source Ecosystem. The prosperity of LLMs is inherently linked to the open-source community. The techniques for effectively leveraging the open-source ecosystem around LLMs are still underdeveloped. Open-source contributions should incorporate collaborative risk assessments and secure coding practices to prevent vulnerabilities. Additionally, establishing ethical guidelines for contributions, such as promoting diversity in training data and addressing bias, can ensure that open-source models are developed responsibly. Furthermore, there is a need for shared benchmarks and evaluation standards that enable developers to compare models transparently, fostering objective performance improvements and driving innovation.

Enhanced Transparency and Traceability. Improved transparency in sourcing especially in data and models during the whole LLM application developing process can build user trust, ensure accountability in LLM development, and foster ethical AI practices. There are three potential approaches. First, creating detailed audit trails for data usage and transformations in training pipelines can enhance accountability and support error tracking. Second, allowing controlled access to different data and model components supports transparency while protecting sensitive information. Third, tracking changes across model versions, from initial models to refined iterations, enables stakeholders to understand the model's evolution and track back any issues related to specific changes.

Regulatory and Standards Development. Establishing standardized practices and regulatory frameworks for the LLM supply chain can help align practices across different providers, ensuring security, compliance, and interoperability. On the one hand, we need to define clear guidelines on data handling, including data anonymization and user consent, which can help prevent privacy infringements. On the other hand, regulatory frameworks could include guidelines for reducing bias, ensuring fairness, etc.

Improved Defense Mechanisms. Developing stronger defenses against adversarial threats can enhance the resilience of LLMs against cyber threats and tampering. Three potential approaches can be considered. First, implementing systems to detect unusual activity or output patterns can help identify and mitigate adversarial attacks effectively. Second, utilizing secure delivery channels from an identified contributor can prevent untrusted modifications to LLMs. Third, establishing automated models or library

updates can address vulnerabilities in LLM applications promptly, reducing the risk of exploitation.

7 Conclusions

The rapid growth of large language models has transformed numerous industries, creating an intricate supply chain that may incur potential risks. This paper presents a comprehensive overview of the LLM supply chain. We identify and categorize the risks inherent in this supply chain, framing them through stakeholders, risky actions, risk types, and specific supply chain components. Additionally, we provide a taxonomy of mitigation strategies, offering actionable guidance for stakeholders seeking to navigate and secure the LLM supply chain. We also highlight emerging challenges and opportunities in securing the LLM supply chain, aiming to inspire further research into robust defenses and proactive security measures.

References

[1] Aydin Abadi, Vishnu Asutosh Dasu, and Sumanta Sarkar. 2024. Privacy-Preserving Data Deduplication for Enhancing Federated Learning of Language Models. *arXiv preprint arXiv:2407.08152* (2024).

[2] NM AbdElnapi, Fatma A Omara, and Nahla F Omran. 2016. A hybrid hashing security algorithm for data storage on cloud computing. *International Journal of Computer Science and Information Security* 14, 4 (2016).

[3] Eugene Bagdasaryan, Andreas Veit, Yiqing Hua, Deborah Estrin, and Vitaly Shmatikov. 2020. How to backdoor federated learning. In *International conference on artificial intelligence and statistics*. 2938–2948.

[4] Fred Bals. 2024. What is the Xz Utils Backdoor : Everything you need to know about the supply chain attack. Retrieved Aug 16, 2024 from https://www.synopsys.com/blogs/software-security/xz-utils-backdoor-supply-chain-attack.html

[5] Tim Baumgärtner, Yang Gao, Dana Alon, and Donald Metzler. 2024. Best-of-Venom: Attacking RLHF by Injecting Poisoned Preference Data. *arXiv preprint arXiv:2404.05530* (2024).

[6] BlackDuck. 2002. BlackDuck. Retrieved May 30, 2024 from https://www.blackduck.com/software-composition-analysis-tools/black-duck-sca.html

[7] bloomberg. 2024. Introducing BloombergGPT, Bloomberg's 50-billion parameter large language model, purpose-built from scratch for finance. Retrieved May 30, 2024 from https://www.bloomberg.com/company/press/bloomberggpt-50-billion-parameter-llm-tuned-finance/

[8] Xiangrui Cai, Haidong Xu, Sihan Xu, Ying Zhang, et al. 2022. Badprompt: Backdoor attacks on continuous prompts. *Advances in Neural Information Processing Systems* 35 (2022), 37068–37080.

[9] Alan Cao and Brendan Dolan-Gavitt. 2022. What the Fork? Finding and Analyzing Malware in GitHub Forks. In *Proceedings of the Network and Distributed System Security Symposium*, Vol. 22.

[10] Nicholas Carlini, Daniel Paleka, Krishnamurthy Dj Dvijotham, Thomas Steinke, Jonathan Hayase, A Feder Cooper, Katherine Lee, Matthew Jagielski, Milad Nasr, Arthur Conmy, et al. 2024. Stealing part of a production language model. *arXiv preprint arXiv:2403.06634* (2024).

[11] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. 2021. Extracting training data from large language models. In *30th USENIX Security Symposium*. 2633–2650.

[12] CheckPoint. 2024. PyPI Inundated by Malicious Typosquatting Campaign. Retrieved May 30, 2024 from https://blog.checkpoint.com/securing-the-cloud/pypi-inundated-by-malicious-typosquatting-campaign/

[13] Jun Chen, Deyao Zhu, Xiaoqian Shen, Xiang Li, Zechun Liu, Pengchuan Zhang, Raghuraman Krishnamoorthi, Vikas Chandra, Yunyang Xiong, and Mohamed Elhoseiny. 2023. Minigpt-v2: large language model as a unified interface for vision-language multi-task learning. *arXiv preprint arXiv:2310.09478* (2023).

[14] Sizhe Chen, Julien Piet, Chawin Sitawarin, and David Wagner. 2024. StruQ: Defending against prompt injection with structured queries. *arXiv preprint arXiv:2402.06363* (2024).

[15] Tianyu Chen, Hangbo Bao, Shaohan Huang, Li Dong, Binxing Jiao, Daxin Jiang, Haoyi Zhou, Jianxin Li, and Furu Wei. 2022. The-x: Privacy-preserving transformer inference with homomorphic encryption. *arXiv preprint arXiv:2206.00216* (2022).

[16] Xiaoyi Chen, Ahmed Salem, Dingfan Chen, Michael Backes, Shiqing Ma, Qingni Shen, Zhonghai Wu, and Yang Zhang. 2021. Badnl: Backdoor attacks against

nlp models with semantic-preserving improvements. In *Proceedings of the 37th Annual Computer Security Applications Conference*. 554–569.

[17] Stefano Chierici. 2022. Analysis on Docker Hub malicious images: Attacks through public container images. Retrieved Aug 16, 2024 from https://sysdig.com/blog/analysis-of-supply-chain-attacks-through-public-docker-images/

[18] Google Cloud. 2024. Vertex AI. Retrieved May 30, 2024 from https://cloud.google.com/vertex-ai/docs/start/introduction-unified-platform

[19] Tianyu Cui, Yanling Wang, Chuanpu Fu, Yong Xiao, Sijia Li, Xinhao Deng, Yunpeng Liu, Qinglin Zhang, Ziyi Qiu, Peiyang Li, et al. 2024. Risk taxonomy, mitigation, and assessment benchmarks of large language model systems. *arXiv preprint arXiv:2401.05778* (2024).

[20] Xuanming Cui, Alejandro Aparcedo, Young Kyun Jang, and Ser-Nam Lim. 2024. On the robustness of large multimodal models against image adversarial attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 24625–24634.

[21] Gelei Deng, Yi Liu, Yuekang Li, Kailong Wang, Ying Zhang, Zefeng Li, Haoyu Wang, Tianwei Zhang, and Yang Liu. 2024. Masterkey: Automated jailbreaking of large language model chatbots. In *Proc. ISOC NDSS*.

[22] Erik Derner, Kristina Batistič, Jan Zahálka, and Robert Babuška. 2023. A security risk taxonomy for large language models. *arXiv preprint arXiv:2311.11415* (2023).

[23] Ruian Duan, Omar Alrawi, Ranjita Pai Kasturi, Ryan Elder, Brendan Saltaformaggio, and Wenke Lee. 2020. Towards measuring supply chain attacks on package managers for interpreted languages. In *Proceedings of the 28th Annual Network and Distributed System Security Symposium*.

[24] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. 2006. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography: Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006. Proceedings 3*. 265–284.

[25] Cynthia Dwork, Aaron Roth, et al. 2014. The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science* 9, 3–4 (2014), 211–407.

[26] Aysan Esmradi, Daniel Wankit Yip, and Chun Fai Chan. 2023. A comprehensive survey of attack techniques, implementation, and mitigation strategies in large language models. In *International Conference on Ubiquitous Security*. 76–95.

[27] GitHub eugeneyan. 2024. Open LLMs. Retrieved May 30, 2024 from https://github.com/eugeneyan/open-llms

[28] Hugging Face. 2024. Hugging Face. Retrieved May 30, 2024 from https://huggingface.co

[29] Angela Fan, Beliz Gokkaya, Mark Harman, Mitya Lyubarskiy, Shubho Sengupta, Shin Yoo, and Jie M Zhang. 2023. Large language models for software engineering: Survey and open problems. *arXiv preprint arXiv:2310.03533* (2023).

[30] Python Software Foundation. 2024. PyPI. Retrieved May 30, 2024 from https://pypi.org

[31] Laura French. 2024. Are your GitHub Action artifacts leaking tokens? Retrieved Aug 16, 2024 from https://www.scmagazine.com/news/are-your-github-action-artifacts-leaking-tokens

[32] GitHub. 2024. GitHub. Retrieved May 30, 2024 from https://github.com

[33] GitHub. 2024. *GPTCache*. Retrieved April 20, 2024 from https://github.com/zilliztech/GPTCache

[34] GitHub. 2024. *Promptify*. Retrieved April 20, 2024 from https://github.com/promptslab/Promptify

[35] GitHub. 2024. *Zapier*. Retrieved April 20, 2024 from https://zapier.com/blog/gpt-assistant/

[36] Danielle Gonzalez, Thomas Zimmermann, Patrice Godefroid, and Max Schäfer. 2021. Anomalicious: Automated detection of anomalous and potentially malicious commits on github. In *Proceedings of the 43rd International Conference on Software Engineering: Software Engineering in Practice*.

[37] Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, 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. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*. 79–90.

[38] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles L Isbell, and Andrea L Thomaz. 2013. Policy shaping: Integrating human feedback with reinforcement learning. *Advances in neural information processing systems* (2013).

[39] Samyak Gupta, Yangsibo Huang, Zexuan Zhong, Tianyu Gao, Kai Li, and Danqi Chen. 2022. Recovering private text in federated learning of language models. *Advances in neural information processing systems* (2022), 8130–8143.

[40] Cheng Huang, Nannan Wang, Ziyan Wang, Siqi Sun, Lingzi Li, Junren Chen, Qianchong Zhao, Jiaxuan Han, Zhen Yang, and Lei Shi. 2024. DONAPI: Malicious NPM Packages Detector using Behavior Sequence Knowledge Mapping. In *Proceedings of the 33rd USENIX Security Symposium*.

[41] Huiqing Huang, Shouzhi Yang, and Ruisong Ye. 2020. Efficient symmetric image encryption by using a novel 2D chaotic system. *IET Image Processing* 14, 6 (2020), 1157–1163.

[42] Xiaoling Huang, Youxia Dong, Hongyong Zhu, and Guodong Ye. 2022. Visually asymmetric image encryption algorithm based on SHA-3 and compressive sensing by embedding encrypted image. *Alexandria Engineering Journal* 61, 10

(2022), 7637–7647.

[43] Yiheng Huang, Ruisi Wang, Wen Zheng, Zhuotong Zhou, Susheng Wu, Shulin Ke, Bihuan Chen, Shan Gao, and Xin Peng. 2024. SpiderScan: Practical Detection of Malicious NPM Packages Based on Graph-Based Behavior Modeling and Matching. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*.

[44] Yujin Huang, Terry Yue Zhuo, Qiongkai Xu, Han Hu, Xingliang Yuan, and Chunyang Chen. 2023. Training-free lexical backdoor attacks on language models. In *Proceedings of the ACM Web Conference 2023*. 2198–2208.

[45] Thomas Hunter II. 2018. Compromised npm Package: event-stream. Retrieved May 30, 2024 from https://medium.com/intrinsic-blog/compromised-npm-package-event-stream-d47d08605502

[46] Umar Iqbal, Tadayoshi Kohno, and Franziska Roesner. 2023. LLM Platform Security: Applying a Systematic Evaluation Framework to OpenAI's ChatGPT Plugins. *arXiv preprint arXiv:2309.10254* (2023).

[47] Matthew Jagielski, Om Thakkar, Florian Tramèr, Daphne Ippolito, Katherine Lee, Nicholas Carlini, Eric Wallace, Shuang Song, Abhradeep Thakurta, Nicolas Papernot, and Chiyuan Zhang. 2023. Measuring Forgetting of Memorized Training Examples. arXiv:2207.00099 [cs.LG] https://arxiv.org/abs/2207.00099

[48] Wenxin Jiang, Nikolas Synovic, Matt Hyatt, Taylor R Schorlemmer, Rohan Sethi, Yung-Hsiang Lu, George K Thiruvathukal, and James C Davis. 2023. An empirical study of pre-trained model reuse in the hugging face deep learning model registry. In *2023 IEEE/ACM 45th International Conference on Software Engineering*. 2463–2475.

[49] Daniel Kang, Xuechen Li, Ion Stoica, Carlos Guestrin, Matei Zaharia, and Tatsunori Hashimoto. 2024. Exploiting programmatic behavior of llms: Dual-use through standard security attacks. In *2024 IEEE Security and Privacy Workshops*. 132–143.

[50] John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2023. A watermark for large language models. In *International Conference on Machine Learning*. 17061–17084.

[51] Eduard Kovacs. 2024. *ChatGPT Plugin Vulnerabilities Exposed Data, Accounts*. Retrieved July 24, 2024 from https://www.securityweek.com/chatgpt-plugin-vulnerabilities-exposed-data-accounts/

[52] Eduard Kovacs. 2024. *Simple Attack Allowed Extraction of ChatGPT Training Data*. Retrieved July 24, 2024 from https://www.securityweek.com/simple-attack-allowed-extraction-of-chatgpt-training-data/

[53] Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. 2023. Robust distortion-free watermarks for language models. *arXiv preprint arXiv:2307.15593* (2023).

[54] Ashutosh Kumar, Sagarika Singh, Shiv Vignesh Murty, and Swathy Ragupathy. 2024. The ethics of interaction: Mitigating security threats in llms. *arXiv preprint arXiv:2401.12273* (2024).

[55] labelbox. 2024. *LabelBox*. Retrieved April 20, 2024 from https://labelbox.com/

[56] Piergiorgio Ladisa, Henrik Plate, Matias Martinez, and Olivier Barais. 2023. Sok: Taxonomy of attacks on open-source software supply chains. In *2023 IEEE Symposium on Security and Privacy*. 1509–1526.

[57] Taehyun Lee, Seokhee Hong, Jaewoo Ahn, Ilgee Hong, Hwaran Lee, Sangdoo Yun, Jamin Shin, and Gunhee Kim. 2023. Who wrote this code? watermarking for code generation. *arXiv preprint arXiv:2305.15060* (2023).

[58] Dacheng Li, Rulin Shao, Hongyi Wang, Han Guo, Eric P Xing, and Hao Zhang. 2022. Mpcformer: fast, performant and private transformer inference with mpc. *arXiv preprint arXiv:2211.01452* (2022).

[59] Suyi Li, Yong Cheng, Wei Wang, Yang Liu, and Tianjian Chen. 2020. Learning to detect malicious clients for robust federated learning. *arXiv preprint arXiv:2002.00211* (2020).

[60] Shaofeng Li, Hui Liu, Tian Dong, Benjamin Zi Hao Zhao, Minhui Xue, Haojin Zhu, and Jialiang Lu. 2021. Hidden backdoors in human-centric language models. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. 3123–3140.

[61] Hairong Lin, Chunhua Wang, Jingru Sun, Xin Zhang, Yichuang Sun, and Herbert HC Iu. 2023. Memristor-coupled asymmetric neural networks: Bionic modeling, chaotic dynamics analysis and encryption application. *Chaos, Solitons & Fractals* 166 (2023), 112905.

[62] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2024. Visual instruction tuning. *Advances in neural information processing systems* (2024).

[63] Google LLC. 2024. Kaggle. Retrieved May 30, 2024 from https://www.kaggle.com

[64] Yuxin Ma, Tiankai Xie, Jundong Li, and Ross Maciejewski. 2019. Explaining vulnerabilities to adversarial machine learning through visual analytics. *IEEE transactions on visualization and computer graphics* (2019).

[65] Justus Mattern, Fatemehsadat Mireshghallah, Zhijing Jin, Bernhard Schoelkopf, Mrinmaya Sachan, and Taylor Berg-Kirkpatrick. 2023. Membership Inference Attacks against Language Models via Neighbourhood Comparison. In *The 61st Annual Meeting Of The Association For Computational Linguistics*.

[66] Microsoft. 2022. sbom-tool. Retrieved May 30, 2024 from https://github.com/microsoft/sbom-tool

[67] Microsoft. 2024. Visual Studio Marketplace. Retrieved May 30, 2024 from https://marketplace.visualstudio.com/

[68] MITRE. 2021. Spamming ML System with Chaff Data. Retrieved May 30, 2024 from https://atlas.mitre.org/techniques/AML.T0046

[69] Adam Murray. 2020. The $100 Million Court Case for Open Source License Compliance. Retrieved May 30, 2024 from https://www.mend.io/blog/the-100-million-case-for-open-source-license-compliance/#track-your-licenses-%E2%80%93-or-risk-litigation

[70] Seth Neel and Peter Chang. 2023. Privacy issues in large language models: A survey. *arXiv preprint arXiv:2312.06717* (2023).

[71] The Hacker News. 2018. Password-Guessing Was Used to Hack Gentoo Linux Github Account. Retrieved May 30, 2024 from https://thehackernews.com/2018/07/github-hacking-gentoo-linux.html

[72] The Hacker News. 2022. 10 Credential Stealing Python Libraries Found on PyPI Repository. Retrieved May 30, 2024 from https://thehackernews.com/2022/08/10-credential-stealing-python-libraries.html

[73] Newsroom. 2024. *Third-Party ChatGPT Plugins Could Lead to Account Takeovers*. Retrieved July 24, 2024 from https://thehackernews.com/2024/03/third-party-chatgpt-plugins-could-lead.html

[74] nvidia GitHub. 2024. *TensorRT*. Retrieved April 20, 2024 from https://nvidia.github.io/TensorRT-Model-Optimizer/guides/1_quantization.html

[75] onnx. 2024. *Onnx*. Retrieved April 20, 2024 from https://onnx.ai/

[76] OpenAI. 2024. *Introducing the GPT Store*. Retrieved July 24, 2024 from https://openai.com/index/introducing-the-gpt-store/

[77] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems* 35 (2022), 27730–27744.

[78] Mustafa Safa Ozdayi, Charith Peris, Jack FitzGerald, Christophe Dupuy, Jimit Majmudar, Haidar Khan, Rahil Parikh, and Rahul Gupta. 2023. Controlling the extraction of memorized data from large language models via prompt-tuning. *arXiv preprint arXiv:2305.11759* (2023).

[79] Xudong Pan, Mi Zhang, Shouling Ji, and Min Yang. 2020. Privacy risks of general-purpose language models. In *2020 IEEE Symposium on Security and Privacy*. 1314–1331.

[80] Xudong Pan, Mi Zhang, Beina Sheng, Jiaming Zhu, and Min Yang. 2022. Hidden trigger backdoor attack on {NLP} models via linguistic style manipulation. In *Proceedings of the 31st USENIX Security Symposium*. 3611–3628.

[81] Rahul Pankajakshan, Sumitra Biswal, Yuvaraj Govindarajulu, and Gilad Gressel. 2024. Mapping LLM Security Landscapes: A Comprehensive Stakeholder Risk Assessment Proposal. *arXiv preprint arXiv:2403.13309* (2024).

[82] Rahil Parikh, Christophe Dupuy, and Rahul Gupta. 2022. Canary Extraction in Natural Language Understanding Models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*. 552–560.

[83] Debian Project. 2024. Debian Packages. Retrieved May 30, 2024 from https://www.debian.org/distrib/packages

[84] PyPI. 2023. 2FA Requirement for PyPI begins 2024-01-01. Retrieved May 30, 2024 from https://blog.pypi.org/posts/2023-12-13-2fa-enforcement/

[85] PyTorch. 2022. Compromised PyTorch-nightly dependency chain between December 25th and December 30th. Retrieved May 30, 2024 from https://pytorch.org/blog/compromised-nightly-dependency/

[86] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* (2019), 9.

[87] Alexander Robey, Eric Wong, Hamed Hassani, and George J Pappas. 2023. Smoothllm: Defending large language models against jailbreaking attacks. *arXiv preprint arXiv:2310.03684* (2023).

[88] Md Omar Faruk Rokon, Risul Islam, Ahmad Darki, Evangelos E. Papalexakis, and Michalis Faloutsos. 2020. SourceFinder: Finding malware Source-Code from publicly available repositories in GitHub. In *Proceedings of the 23rd International Symposium on Research in Attacks, Intrusions and Defenses*. 149–163.

[89] ODSC Open Data Science. 2024. 2023 Was the Year of Large Language Models: Then and Now. Retrieved May 30, 2024 from https://odsc.medium.com/2023-was-the-year-of-large-language-models-then-and-now-924d34f3b6a9

[90] scmagazine. 2024. *SAP AI Core flaws show risks of training AI in shared environments*. Retrieved July 24, 2024 from https://www.scmagazine.com/news/sap-ai-core-flaws-show-risks-of-training-ai-in-shared-environments?nbd=%7B%7Blead.HumId%7D%7D&nbd_source=mrkto&utm_source=sc-dailyscan&utm_medium=email&mkt_tok=MTg4LVVOWi02NjAAAAGUaWl_tY1A-NhfVVQomaBYG1UhBIqdj_oZrzANo6xuZnI1EVvKBl4TnmU-Y1RRZU4vYvsphD9f7NNGM3wxsvxl41hXY4niWZFW6hSx-CHM-2Av

[91] Amazon Web Services. 2024. Deep Learning AMI. Retrieved May 30, 2024 from https://aws.amazon.com/machine-learning/amis/

[92] Shawn Shan, Wenxin Ding, Josephine Passananti, Stanley Wu, Haitao Zheng, and Ben Y Zhao. 2024. Nightshade: Prompt-Specific Poisoning Attacks on Text-to-Image Generative Models. In *2024 IEEE Symposium on Security and Privacy*. 212–212.

[93] Erfan Shayegani, Yue Dong, and Nael Abu-Ghazaleh. 2023. Jailbreak in pieces: Compositional adversarial attacks on multi-modal language models. In *Proceedings of the 12th International Conference on Learning Representations*.

[94] Xinyue Shen, Zeyuan Chen, Michael Backes, Yun Shen, and Yang Zhang. 2023. " do anything now": Characterizing and evaluating in-the-wild jailbreak prompts on large language models. *arXiv preprint arXiv:2308.03825* (2023).

[95] Manli Shu, Jiongxiao Wang, Chen Zhu, Jonas Geiping, Chaowei Xiao, and Tom Goldstein. 2023. On the exploitability of instruction tuning. *Advances in Neural Information Processing Systems* (2023).

[96] Ilia Shumailov, Yiren Zhao, Daniel Bates, Nicolas Papernot, Robert Mullins, and Ross Anderson. 2021. Sponge examples: Energy-latency attacks on neural networks. In *2021 IEEE European symposium on security and privacy*. 212–231.

[97] Wai Man Si, Michael Backes, and Yang Zhang. 2023. Mondrian: Prompt abstraction attack against large language models for cheaper API pricing. *arXiv preprint arXiv:2308.03558* (2023).

[98] Sonatype. 2008. Automate your dependency management. Retrieved May 30, 2024 from https://www.sonatype.com/sonatype-developer

[99] Robin Staab, Mark Vero, Mislav Balunovic, and Martin Vechev. 2023. Beyond Memorization: Violating Privacy via Inference with Large Language Models. In *Proceedings of the 12th International Conference on Learning Representations*.

[100] Ludwig-Ferdinand Stumpp. 2024. *Achieving Code Execution in MathGPT via Prompt Injection*. Retrieved July 24, 2024 from https://atlas.mitre.org/studies/AML.CS0016/

[101] Zhensu Sun, Xiaoning Du, Xiapu Luo, Fu Song, David Lo, and Li Li. 2024. FDI: Attack Neural Code Generation Systems through User Feedback Channel. *arXiv preprint arXiv:2408.04194* (2024).

[102] Zhensu Sun, Xiaoning Du, Fu Song, and Li Li. 2023. Codemark: Imperceptible watermarking for code datasets against neural code completion models. In *Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. 1561–1572.

[103] Xuchen Suo. 2024. Signed-Prompt: A new approach to prevent prompt injection attacks against LLM-integrated applications. *arXiv preprint arXiv:2401.07612* (2024).

[104] Ruixiang Tang, Qizhang Feng, Ninghao Liu, Fan Yang, and Xia Hu. 2023. Did you train on my dataset? towards public dataset protection with cleanlabel backdoor watermarking. *ACM SIGKDD Explorations Newsletter* 25, 1 (2023), 43–53.

[105] LLM Red Team. 2024. *Free API by LLM Red Team*. Retrieved July 24, 2024 from https://github.com/LLM-Red-Team/free-api/

[106] Aparna Tomar, Diksha Jeena, Preeti Mishra, and Rahul Bisht. 2020. Docker security: A threat model, attack taxonomy and real-time attack scenario of dos. In *10th International Conference on Cloud Computing, Data Science & Engineering*. 150–155.

[107] Jean-Baptiste Truong, Pratyush Maini, Robert J Walls, and Nicolas Papernot. 2021. Data-free model extraction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 4771–4780.

[108] VirusTotal. 2023. VirusTotal. Retrieved May 30, 2024 from https://www.virustotal.com/

[109] Alexander Wan, Eric Wallace, Sheng Shen, and Dan Klein. 2023. Poisoning language models during instruction tuning. In *International Conference on Machine Learning*. 35413–35425.

[110] Yao Wan, Shijie Zhang, Hongyu Zhang, Yulei Sui, Guandong Xu, Dezhong Yao, Hai Jin, and Lichao Sun. 2022. You see what i want you to see: poisoning vulnerabilities in neural code search. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*.

[111] Lean Wang, Wenkai Yang, Deli Chen, Hao Zhou, Yankai Lin, Fandong Meng, Jie Zhou, and Xu Sun. 2023. Towards codable text watermarking for large language models. *arXiv preprint arXiv:2307.15992* (2023).

[112] Shenao Wang, Yanjie Zhao, Xinyi Hou, and Haoyu Wang. 2024. Large language model supply chain: A research agenda. *arXiv preprint arXiv:2404.12736* (2024).

[113] Ziqiu Wang, Jun Liu, Shengkai Zhang, and Yang Yang. 2024. Poisoned LangChain: Jailbreak LLMs by LangChain. *arXiv preprint arXiv:2406.18122* (2024).

[114] Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. 2024. Jailbroken: How does llm safety training fail? *Advances in Neural Information Processing Systems* (2024).

[115] Simon Willison. 2024. *Prompt injection and jailbreaking are not the same thing*. Retrieved July 24, 2024 from https://simonwillison.net/2024/Mar/5/prompt-injection-jailbreaking/

[116] Fangzhou Wu, Ning Zhang, Somesh Jha, Patrick McDaniel, and Chaowei Xiao. 2024. A new era in llm security: Exploring security concerns in real-world llm-based systems. *arXiv preprint arXiv:2402.18649* (2024).

[117] Sophie Xhonneux, Alessandro Sordoni, Stephan Günnemann, Gauthier Gidel, and Leo Schwinn. 2024. Efficient adversarial training in llms with continuous attacks. *arXiv preprint arXiv:2405.15589* (2024).

[118] Weilin Xu, David Evans, and Yanjun Qi. 2017. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155* (2017).

[119] Zhou Yang, Zhipeng Zhao, Chenyu Wang, Jieke Shi, Dongsun Kim, DongGyun Han, and David Lo. 2023. What do code models memorize? an empirical study on large language models of code. *arXiv preprint arXiv:2308.09932* (2023).

[120] Hongwei Yao, Jian Lou, and Zhan Qin. 2024. Poisonprompt: Backdoor attack on prompt-based large language models. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing*. 7745–7749.

[121] Yifan Yao, Jinhao Duan, Kaidi Xu, Yuanfang Cai, Zhibo Sun, and Yue Zhang. 2024. A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. *High-Confidence Computing* (2024), 100211.

[122] Weichen Yu, Tianyu Pang, Qian Liu, Chao Du, Bingyi Kang, Yan Huang, Min Lin, and Shuicheng Yan. 2023. Bag of tricks for training data extraction from language models. In *International Conference on Machine Learning*. 40306–40320.

[123] Xiang Yue, Huseyin A Inan, Xuechen Li, Girish Kumar, Julia McAnallen, Hoda Shajari, Huan Sun, David Levitan, and Robert Sim. 2022. Synthetic text generation with differential privacy: A simple and practical recipe. *arXiv preprint arXiv:2210.14348* (2022).

[124] Junan Zhang, Kaifeng Huang, Bihuan Chen, Chong Wang, Zhenhao Tian, and Xin Peng. 2023. Malicious Package Detection in NPM and PyPI using a Single Model of Malicious Behavior Sequence. *arXiv preprint arXiv:2309.02637* (2023).

[125] Zaixi Zhang, Xiaoyu Cao, Jinyuan Jia, and Neil Zhenqiang Gong. 2022. Fldetector: Defending federated learning against model poisoning attacks via detecting malicious clients. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2545–2555.

[126] Zhexin Zhang, Jiaxin Wen, and Minlie Huang. 2023. ETHICIST: Targeted Training Data Extraction Through Loss Smoothed Soft Prompting and Calibrated Confidence Estimation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*. 12674–12687.

[127] Jian Zhao, Shenao Wang, Yanjie Zhao, Xinyi Hou, Kailong Wang, Peiming Gao, Yuanchao Zhang, Chen Wei, and Haoyu Wang. 2024. Models Are Codes: Towards Measuring Malicious Code Poisoning Attacks on Pre-trained Model Hubs. *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering* (2024).

[128] Yanjie Zhao, Xinyi Hou, Shenao Wang, and Haoyu Wang. 2024. Llm app store analysis: A vision and roadmap. *arXiv preprint arXiv:2404.12737* (2024).

[129] Tong Zhou, Yukui Luo, Shaolei Ren, and Xiaolin Xu. 2023. NNSplitter: an active defense solution for DNN model via automated weight obfuscation. In *International Conference on Machine Learning*. 42614–42624.

[130] Andy Zou, Zifan Wang, J Zico Kolter, and Matt Fredrikson. 2023. Universal and transferable adversarial attacks on aligned language models. *arXiv preprint arXiv:2307.15043* (2023).

[131] Wei Zou, Runpeng Geng, Binghui Wang, and Jinyuan Jia. 2024. Poisonedrag: Knowledge poisoning attacks to retrieval-augmented generation of large language models. *arXiv preprint arXiv:2402.07867* (2024).