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

KAIFENG HUANG[*], Tongji University, China
BIHUAN CHEN[†], Fudan University, China
YOU LU[‡†], Fudan University, China
SUSHENG WU[†‡], Fudan University, China
DINGJI WANG[†], Fudan University, China
YIHENG HUANG[†], Fudan University, China
HAOWEN JIANG[†], Fudan University, China
ZHUOTONG ZHOU[†], Fudan University, China
JUNMING CAO[†], Fudan University, China
XIN PENG[†], Fudan University, China

Large language models (LLMs) have sparked significant impact with regard to both intelligence and productivity. Numerous enterprises have integrated LLMs into their applications to solve their own domain-specific tasks. However, integrating LLMs into specific scenarios is a systematic process that involves substantial components, which are collectively referred to as the *LLM supply chain*. A comprehensive understanding of LLM supply chain composition, as well as the relationships among its components, is crucial for enabling effective mitigation measures for different related risks. A comprehensive understanding of LLM supply chain composition and the relationships among its components, particularly from a security perspective, is crucial for enabling effective mitigation of associated risks. While existing literature has explored various risks associated with LLMs, there remains a notable gap in systematically characterizing the LLM supply chain from the dual perspectives of contributors and consumers. In this work, we develop a structured taxonomy encompassing risk types, risky actions, and corresponding mitigations across different stakeholders and components of the supply chain. We believe that a thorough review of the LLM supply chain composition, along with its inherent risks and mitigation measures, 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.

CCS Concepts: • **Security and privacy** → *Software security engineering*; • **Software and its engineering** → *Software libraries and repositories*; *Risk management*.

[*]K. Huang is with the School of Computer Science and Technology, Tongji University, China.
[†]B. Chen, Y. Lu, S. Wu, D. Wang, Y. Huang, H. Jiang, Z. Zhou, J. Cao, X. Peng. are with the College of Computer Science and Artificial Intelligence and Shanghai Key Laboratory of Data Science, Fudan University, China.
[‡]Y. Lu and S. Wu are the corresponding authors.

Authors' Contact Information: 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.

Manuscript submitted to ACM

Additional Key Words and Phrases: large language model supply chain, open source, large language model risk

ACM Reference Format:
Kaifeng Huang, Bihuan Chen, You Lu, Susheng Wu, Dingji Wang, Yiheng Huang, Haowen Jiang, Zhuotong Zhou, Junming Cao, and Xin Peng. 2026. Lifting the Veil on Composition, Risks, and Mitigations of the Large Language Model Supply Chain. 1, 1 (July 2026), 41 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

The large language models (LLMs) have sparked unprecedented discussion about the power of generative language models. There has been a significant surge in the development and introduction of LLMs, 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 [14, 68, 253].

Integrating LLMs involves more than just utilizing the models themselves. On the one hand, although some applications may directly employ commercial LLMs, they still require additional solutions such as pre-processing (*e.g.,* Promptify [89] for prompt engineering, GPTCache [86] for reducing LLM API call expenses), etc. On the other hand, businesses may prefer open-source LLMs to enable flexible customizations, which demand more sophisticated components, during stages such as fine-tuning (*e.g.,* Labelbox [144]), model conversion (*e.g.,* Onnx [216]), quantization (*e.g.,* TensorRT [213]), etc. Consequently, there exists a multitude of sub-components that are transitively depended upon. Furthermore, the composition types in the *LLM supply chain* are more varied than in the traditional software supply chain. Overall, leveraging LLMs is a systematic process that entails numerous components and composition types, collectively referred to as the *LLM supply chain*.

Accordingly, we are confronted with a fresh LLM supply chain ecosystem, where the attack surface has expanded but remains understudied, and the number of participants is rapidly increasing. While the LLM supply chain shares similarities with the well-recognized open-source software supply chain [146], it also exhibits its uniqueness. For instance, attackers can exploit publicly accessible LLMs to extract sensitive information (*e.g.,* privacy leakage) or fine-tune open-source LLMs [65] to embed malicious instructions (*e.g.,* backdoor attacks).

We provide a novel and comprehensive definition for the LLM supply chain, as shown in Figure 1. It encompasses the ecosystem and sequence of processes integral to developing, training, deploying, and distributing applications that leverage LLMs, including:

- *Stakeholders in various roles. e.g.,* developers contributing to the artifacts, organizations managing the platforms, or vendors providing cloud computing services.
- *Upstream artifacts in multiple forms. e.g.,* plaintext source code, binary executables, or serialized models.
- *Diverse supply relationships. e.g.,* augmentation relationships for data, cloning relationships for code, and merging relationships for models.
- *Development stages for various purposes. e.g.,* data preprocessing, model pre-training, fine-tuning, integration, and monitoring.

While existing literature discusses risks associated with LLMs, there rarely exists a comprehensive work that outlines the LLM supply chain from the perspective of contributors and consumers. We aim to answers the following research questions:

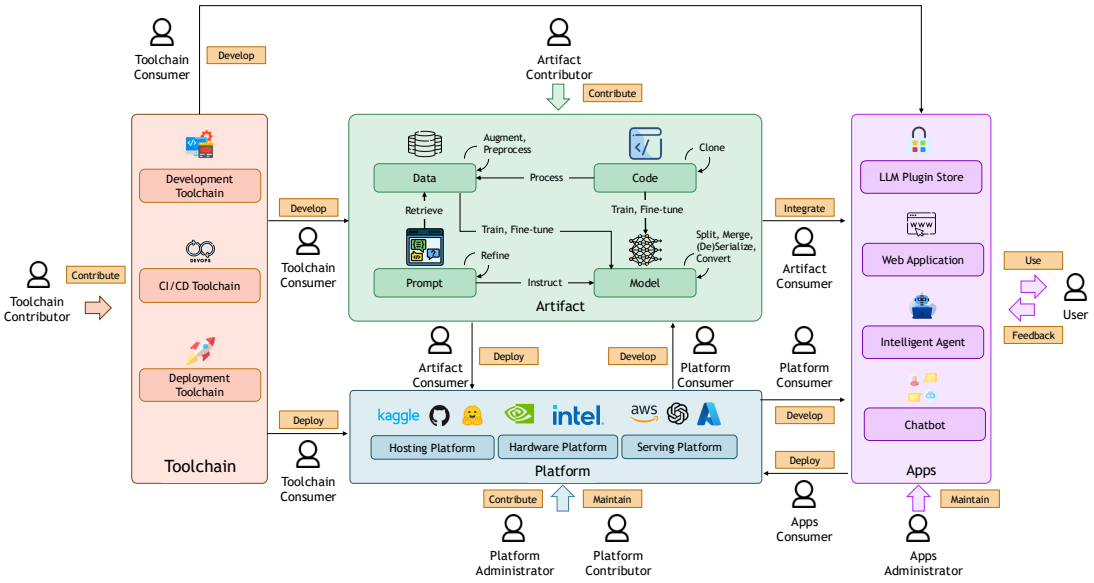

Fig. 1. Overview of the LLM Supply Chain

RQ1 Composition Analysis: What new components accompanied with their defining characteristics, emerge in the LLM supply chain, compared with the software supply chain?

RQ2 Risk Analysis: How can we characterize the risks posed by the LLM supply chain? What distinguishes them from the software supply chain?

RQ3 Mitigation Analysis: How can we mitigate the risks in the LLM supply chain?

To study **RQ1**, 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 that of Wang et al. [325] by placing greater emphasis on the fundamental components and the supply relationships among them.

To explore **RQ2**, we develop taxonomies of risk types, and risky actions related to various LLM supply chain stakeholders and components. Specifically, we first holistically collect 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 risky contributors, downstream risky users, and risky administrators. Furthermore, to study **RQ3**, we list the types of various mitigation in response to the LLM supply chain risks.

Our work 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 contributions of our paper are as follows:

- We present a comprehensive overview of the *LLM supply chain*, including components, stakeholders, and composition relations.
- We develop a detailed taxonomy of risk and mitigations with regard to stakeholders, risk types and supply chain components, providing actionable guidance for practitioners involved in the LLM supply chain.
- We envision future challenges and benefits in securing the LLM supply chain.

2 Related Surveys

Various security-related aspects of LLMs are discussed in existing surveys. For ease of comparison, we focus our topics on *attacks and risks*, *defenses and mitigations*. Additionally, we unify 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 seven survey papers [44, 51, 140, 229, 247, 338, 362], 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 [44, 51, 64, 140, 153, 205, 229, 247, 362] such as membership inference attacks [59, 163], privacy leakage (e.g., PPI) attacks, gradient leakage attacks, model inversion attacks, and attribute inference attacks. Besides, *Prompt Attacks* is covered in eight papers [44, 51, 64, 140, 153, 229, 267, 362], including prompt injection attack and jailbreaking attack. A few papers [44, 229, 338, 362] mention *Toolchain Attacks*. Specifically, Pankajakshan et al.[229] and Yao et al.[362] mention supply chain vulnerabilities but do not elaborate on the components of the supply chain. Wu et al. [338] highlight malicious web tool misuse and cross-session access, which are closely related but represent only a subset of toolchain attacks. Cui et al. [44] identify software toolchain modules as programming language runtime, CI/CD environment, deep learning framework and pre-processing tools. Finally, Neel et al. [205] particularly addressed the issue of *Copyright Risks* in their survey.

Defenses and Mitigation. Defenses and mitigation are suggested in accordance with the corresponding attacks and risks. For *Prompt Attacks*, existing works propose *Input Sanitization* [116, 229]. To mitigate *Output Risk*, approaches like *Output Sanitization* are recommended by several studies [44, 140, 362]. For *Data Attacks*, the literature suggests techniques such as *Data Cleaning* [362], *Data De-duplication* [205], and *Data Sanitization* [64]. 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 makes the following contributions, highlighting its novelty and distinguishing it from existing surveys. First, we conduct a comprehensive survey of both academic literature and online resources with a focus on the *open-source LLM supply chain*, which has not been systematically collected and analyzed. Second, we provide a detailed review of the key stakeholders, components, and their supply relationships within the LLM ecosystem, offering a more structured and process-oriented perspective than prior work [325]. Third, we present a structured illustration of risk scenarios, linking *risky stakeholders* to *risky actions*, *supply chain components*, and resulting *risk types*, from the perspective of three major stakeholder roles. The structured illustration of risk scenarios helps readers situate themselves within the LLM supply chain ecosystem and gain a clearer understanding of the associated risks. Finally, we summarize mitigation strategies mapped to these risk scenarios, providing actionable guidance for managing LLM supply chain risks.

3 Methodology

Following the systematic literature review guidelines [79], we adopt a scientific and systematic approach for literature collection (see Sec. 3.1), then we perform a taxonomy analysis (see Sec. 3.2). An overview of our review process is presented in Figure 2.

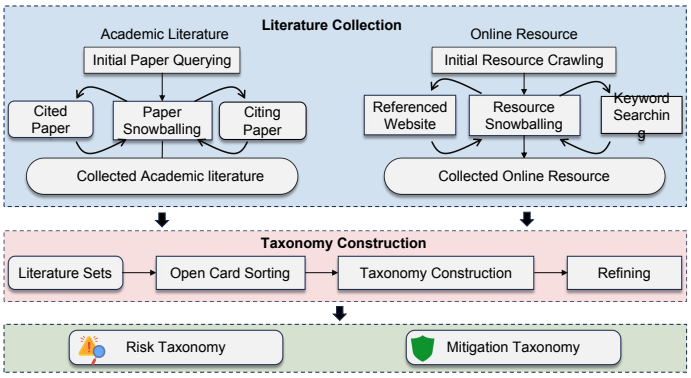

Fig. 2. Approach Overview of Our Literature Review

3.1 Literature Collection

We review the academic literature and online resources to collect an extensive list of risks and mitigations about LLM supply chain.

3.1.1 Academic Literature Collection. We define the temporal scope of our survey to span from the emergence of the earliest relevant publications in this field to the most recent contributions, specifically covering the period from January 2018 to December 2024 (*i.e.,* around 7 years). 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 this process. The detailed steps are as follows.

Initial Paper Querying. We identify potentially relevant papers by conducting searches across major electronic scholarly databases. Specifically, we select ACM Digital Library[1], IEEE Xplore[2], DBLP[3] and Semantic Scholar[4] as our primary data sources, which are popular bibliography databases containing a comprehensive list of research venues in computer science. Then, we define the keywords presented in Listing 1 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).

Listing 1. Search Keywords for Literature Collection

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

Furthermore, we perform a manual assessment to determine the relevance of each candidate paper to the topic of LLM supply chain, excluding those that do not align with the goal of our study. Besides, we exclude short papers with fewer than 2 pages, as they usually contain only preliminary analysis and do not support rigorous or consistent comparison; we also exclude survey and summary papers, which are discussed separately in Sec. 2 for comparison with our work. Consequently, we obtain an initial set of 127 papers.

[1]https://dl.acm.org/
[2]https://ieeexplore.ieee.org/Xplore/home.jsp
[3]https://dblp.org/
[4]https://www.semanticscholar.org/me/research

Paper Snowballing. To mitigate the risk of omitting relevant papers, we apply both backward and forward snowballing techniques [335] on the initial paper set. In the backward snowballing phase, we check the reference lists of these papers to discover new candidates. In forward snowballing phase, we use the "cited by" feature in Google Scholar[5] to identify papers that cite the initial set. This iterative process is repeated until no further relevant papers are found. All newly identified candidate papers are also manually assessed for relevance, resulting in a final set of 238 papers.

3.1.2 Online Resource Collection. In addition to academic literature, we also include online resources covering the same time period. *e.g.,* blog posts and incident reports, which often include real-world attacks or the latest knowledge not yet covered by academic literature. To identify candidate sources, we first employ the same keywords used in the academic literature collection in Google search engine[6]. We exclude sponsored search results, as they are primarily associated with product promotion rather than security reports. From the first three pages of search results, we collect 51 candidate websites. Through manual inspection and assessment, we retain 10 websites. These websites are selected based on their broad coverage, strong reputation in the cybersecurity community, professional content, timely updates, authoritative reporting, and the availability of dedicated blog or news pages for security-related content. For resource collection, we apply these keywords directly in the website search boxes where such functionality is available. For websites without search functionality, we crawl their blog or news sections and filter the content using the same keywords. After retrieving candidate reports from these websites, we apply the following screening procedure. (1) **Relevance screening:** following the procedure in the academic literature collection, the two PhD students independently skim the title, tags, and leading paragraphs to determine whether the content is within our scope of model supply chain security (e.g., risks and mitigations related to data/model artifacts, dependencies, distribution, deployment, and updates), and exclude items that are out-of-scope (e.g., high-level discussions or commercial tool promotions). (2) **Content check:** for the remaining items, we conduct full-text inspection and retain only those that provide verifiable technical details, such as an explicit attack chain, references to CVEs/advisories when applicable, and/or a clear description of affected components (e.g., datasets, model checkpoints, libraries, build/release pipelines, or runtime infrastructure) that establishes a concrete connection to the model supply chain. Any disagreements between the two students are resolved via discussion and, when necessary, adjudicated by a third student.

Table 1 lists the 10 selected websites along with the number of candidate reports collected via keyword searching or crawling, and the final number of reports retained after our two-step screening procedure. In total, we collected 152 candidate reports and retained 31 reports after screening. The excluded reports include, for example, marketing content such as product announcements and vendor advertisements focused on commercial tool capabilities rather than security analysis, high-level discussions about general AI trends (*e.g.,* future of AI, industry predictions) without concrete supply chain security implications, and developer guides providing build tips or configuration instructions for ML tools but lacking coverage of security risks or attack scenarios.

3.2 Taxonomy Construction

We download all the relevant resources and conduct a full-text analysis to inspect and categorize various elements of LLM risks and mitigations from the literature. The labeling process was conducted systematically in two main steps. First, each paper was reviewed, and key content relevant to our research questions was extracted. We applied open

[5]https://scholar.google.com/
[6]https://www.google.com/

Table 1. Online Resource Websites and Report Statistics

Website	Link	Initial	Retained
GitHub Blog	https://github.blog/	29	2
JFrog	https://jfrog.com/blog	7	3
Sonatype	https://www.sonatype.com/blog	13	0
Snyk	https://snyk.io/blog/	3	0
Hacker News	https://thehackernews.com/	32	6
Checkmarx	https://checkmarx.com/blog/	2	1
Check Point	https://blog.checkpoint.com/	12	2
The Register	https://www.theregister.com/security	28	7
Security Intelligence	https://securityintelligence.com/news/	16	6
SC Magazine	https://www.scmagazine.com/	10	4
Total		**152**	**31**

coding to identify distinct concepts and themes (i.e., composition, risks, mitigation) in each section. After extracting concepts, papers were further categorized along multiple dimensions, including the specific risks identified, stakeholders involved, affected supply chain stages, and proposed mitigation strategies. This allowed us to capture both intra-paper details and inter-paper comparisons, ensuring a consistent and structured labeling process across the dataset. Following initial coding, researchers collaboratively organized coded elements into hierarchical tree structures, refined through multiple iterations based on six evaluation criteria (hierarchy, accuracy, comprehensibility, relevance, mutual exclusivity, and coverage). Each iteration involved independent assessment by all researchers, followed by consensus discussion. This process continued until complete agreement was reached on the final taxonomy. We organize tree structures where the root represents the risks, stakeholders, and mitigations. In practice, we typically performed 3 iterative refinement rounds, where each round consisted of independent assessment followed by group discussion. The process terminated when no further structural changes were proposed by any annotator. The six evaluation criteria is listed as follows:

- **Hierarchy.** Evaluate whether the tree's structure is logical, ensuring broader attack categories at higher levels and specific methods at lower levels.
- **Accuracy.** Verify the precision of node descriptions by consulting attack case studies and review discussions. We verify whether each node is explicitly supported by evidence from at least one concrete source, such as documented attack case studies, CVEs, or established security frameworks. Nodes whose descriptions could not be grounded in real-world examples or references were revised.
- **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 (i.e., MITRE ATLAS, OWASP Top 10, and NIST AI RMF).
- **Mutual Exclusivity.** Assess whether each node is distinct from the others.
- **Coverage.** We evaluate coverage by cross-checking each category against existing taxonomies and databases (e.g., MITRE ATLAS, OWASP Top 10 for LLMs, and NIST AI Risk Management Framework) to ensure that all major known risk or mitigation types within the category are represented. Missing techniques identified during this comparison were added in subsequent iterations.

Each node in the structures is assessed by the four annotators independently. All four hold PhD degrees and have relevant expertise in AI security. Two are researchers working in top-tier AI industry organizations, and the other two are faculty members (a professor and an associate professor) at leading universities. During the taxonomy construction and refinement phases, these four researchers independently assessed each node in our hierarchical structures according to the six criteria. The structures are refined and re-evaluated iteratively until all annotators reach a final agreement.

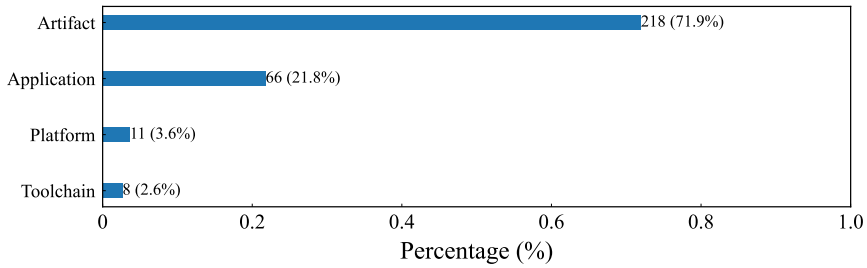

Fig. 3. Distribution of Components that Covered in our Collected Literature

Furthermore, the label is left blank if the literature does not cover specific aspects (*e.g.,* mitigation). Due to unknown biases, inaccuracies, and insensitivity to subtle distinctions in AI tools, which may affect methodological reproducibility, we avoided using AI tools or LLMs for the literature review.

4 Composition of the Large Language Model Supply Chain (RQ1)

We present an overview of the LLM supply chain in Figure 1. In general, we identify four types of *stakeholders*, interacting with specific *components* of the LLM supply chain.

4.1 Stakeholders

We summarize four roles of stakeholders in the LLM supply chain. i.e., *contributors*, *consumers*, *users*, and *administrators*.

Contributors. Person who engaged in development activities and sharing artifacts, such as data, models, prompts, or third-party libraries, etc.

- *Platform Contributors* are responsible for contributing to hosting platforms, hardware platforms, and serving platforms (*i.e.,* Amazon Web Services [258]) and providing stable, isolated, and managed environments.
- *Artifact Contributors* 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.

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 perform interactions with LLM applications (apps) in various forms. Their feedbacks (explicitly or implicitly) may be collected back to the apps.

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

- *Platform Administrators* represent a group of people who manage and oversee the infrastructure and another group who manage repositories of artifacts. They hold the privileges to merge pull requests, manage versioning, and onboard new contributors.
- *App Administrators* are responsible for managing the lifecycle of applications that rely on the artifacts and models within the platform. This includes deploying, configuring, and monitoring applications.

4.2 Components

We summarize four components, including *artifact*, *toolchain*, *platform*, and *apps*. Figure 3 illustrates the distribution of supply chain components covered in the surveyed literature. Notably, the majority of the works (218 papers, 71.9%) focus on risks associated with *artifacts*, followed by *applications* (66 papers, 21.8%). In contrast, significantly fewer studies address risks related to the *platform* (11 papers, 3.6%) and *toolchain* (8 papers, 2.6%). This imbalance is partly attributable to the fact that there are substantially fewer distinct platforms and toolchains compared to models, data, or prompts in the LLM ecosystem.

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

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.

Data. The data can be divided as the corpus for pre-training or fine-tuning, and the knowledge database for retrieval augmented generation.

Model. The model represents a LLM that is trained and fine-tuned. The model may be adjusted or modified through various transformations, including splitting [397], merging [378], serializing or deserializing [103], and quantizing [141] its components, among others, to optimize performance.

Prompt. The prompt is crafted to instruct the model effectively, guiding it on what information to retrieve, process, or generate.

4.2.2 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, and sharing of software packages, models, data, and related resources required for LLM development and deployment [48, 84]. *e.g.,* Data Hosting Platforms (*e.g.,* Kaggle [179]), Model Hosting Platforms (*e.g.,* Hugging Face [67]), Repository Hosting Platforms (*e.g.,* GitHub [85]), Package Registries (*e.g.,* PyPI [74]), Plugin Marketplaces (e.g., Visual Studio Marketplace [196]).

Hardware Platform. These platforms consist of specialized hardware designed to optimize the performance of LLMs (e.g., compute hardware such as NVIDIA's H-series GPUs [211], and GPU interconnection such as NVLink [212]).

Serving Platform. These platforms offer cloud-based services to support the training, deployment, and maintenance of LLMs (e.g., Amazon's AWS cloud computing platform [298, 300]).

4.2.3 Toolchain. The development and deployment of LLMs involve three toolchains: development toolchain, deployment toolchain and CI/CD Toolchain.

Development Toolchain. The development toolchain encompasses the tools and workflows used to train, refine, and optimize the model, as well as to design and build applications around it. Key components include training frameworks [6, 15, 145, 206], model serialization and versioning [180], model conversion and model quantization [150, 399] tools, IDEs [122, 312].

Deployment Toolchain. The deployment toolchain focuses on integrating both the trained model and its associated applications into production environments, and managing them throughout their lifecycle. It automates deployment, monitoring [52], and real-time updates. Core components include cloud services [191, 297] or containerization tools (e.g., Docker) that provide the deployment environment.

CI/CD Toolchain. CI/CD (Continuous Integration and Continuous Delivery) toolchain underpins both development and deployment pipelines by automating integration, testing, and release processes. It ensures that model updates, such as retraining, fine-tuning, or patching, can be rapidly validated and safely rolled out. Key components include source code management systems (e.g., GitHub), automated testing frameworks that validate model correctness and robustness, build and orchestration tools (e.g., Jenkins [120]), and artifact repositories for storing intermediate and production-ready models.

4.2.4 Applications. Applications provide functionalities and services directly used by users. These applications include chatbots [18, 78, 129, 226], web applications, intelligent agents [78, 306, 385], text-generation tools, and more. Application developers integrate large models into real-world products to develop intelligent applications with API interfaces. For example, the plugins on the LLM plugin store [307] 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 [395].

Summary. We examined four key components in the LLM supply chain: artifacts, platforms, toolchains, and applications. These components can be further broken down into four types of artifacts, three categories of platforms, three toolchain types, and four applications. The majority of the works (71.9%) focus on risks associated with *artifacts*, followed by *applications* (21.8%), *platform* (3.6%) and *toolchain* (2.6%). As platform and toolchain components constitute foundational and widely adopted layers of the LLM supply chain, their systematic analysis warrants greater research attention.

4.3 Composition Relations

We discuss the composition relations within the LLM supply chain from the perspectives of declared and undeclared relations. The relation is derived from components across artifacts, toolchains, applications, platforms, and other related elements.

4.3.1 Declared Relations. Declared relations are those that can be directly identified from artifacts such as source code, configuration files (*e.g.*, package manager configuration files), and documentation.

Invocation Relations from Source Code. Declared dependencies are often revealed through import statements or direct method calls. These include class or method imports, method invocations, the use of imported constants, and API invocations.

Dependency Declaration in Config Files. Dependencies explicitly declared in configuration files, such as `package.json` in JavaScript or `pom.xml` in Java, provide clear indications of relationships between components. These files serve as a formal record of required libraries or modules.

Dependency Statements in Documentation. Declared relations can also be derived from documentation, such as API references, SBOMs [284], README files, or inline comments, which explicitly describe dependencies, interactions, or usage guidelines.

4.3.2 Undeclared Relations. Undeclared relations are those embedded during the development and deployment of LLMs. Unlike declared relations, they cannot be **directly** inferred from existing artifacts. Nevertheless, they are an inevitable and integral component of the LLM supply chain.

Training Relations from Training and Fine-tuning Dataset. Data stands as one of the most critical components in training LLMs. The direct dependencies on data are consolidated into the trained weights, which are inherently embedded within the models. While explicit patterns are directly extracted, implicit relations, though not directly extractable, can still be inferred through techniques such as membership inference attacks [59, 71].

Prompt Augmentation Relations from RAG Dataset and Third-party Prompts. The RAG dataset serves as an external knowledge source, enabling LLMs to acquire a more comprehensive understanding of the queried questions. Meanwhile, third-party prompts offer detailed and tailored instructions, making it easier for LLMs to interpret and respond effectively. These elements act as supplementary knowledge sources, providing richer contextual information for LLMs. As a result, we define these relationships as prompt augmentation relations.

Artifact Cloning Relations. In traditional software development, code cloning commonly occurs as a well-known reuse practice. As we transition to the LLM supply chain and encounter new types of artifacts, novel forms of cloning have emerged. For example, to fulfill model cloning, downstream users may split, merge, or convert models, reflecting more dynamic and complex reuse patterns [34, 339]. Moreover, datasets often undergo preprocessing as a standard part of the pipeline.

Invoked Toolchain Relations. Toolchains used during model development and deployment introduce implicit dependencies that are often overlooked. They operate on models, data, and code during development and deployment, but are not dependencies embedded in the resulting artifacts. However, their influence is implicitly present through aspects such as compling optimization strategies, or deployment configurations. Consequently, the invoked toolchains form implicit relations that affect the model behavior, reproducibility, and potential vulnerabilities across the LLM supply chain.

Comparison to traditional software supply chain composition. Traditional software is primarily composed of explicitly packaged and versioned artifacts, such as source code, libraries, binaries, and build tools whose relationships are largely manifested through declared dependencies and configurations. In contrast, the LLM supply chain extends this composition beyond code-centric artifacts to include data-centric and behavior-defining components, such as training and fine-tuning datasets, prompt templates, RAG datasets, as well as model structure and weights. While traditional software composition emphasizes modular reuse through well-defined interfaces and dependency graphs, LLM composition inherently embeds upstream components into opaque artifacts. This fundamental shift in supply chain composition highlights the need for LLM-specific composition analysis that accounts for both explicit and implicitly dependency relations.

Overall, the LLM supply chain, particularly due to its reliance on training data and prompt-based augmentation, exhibits a predominance of undeclared relations [46, 59, 185], as many critical dependencies are implicitly embedded in models, prompts, datasets, and toolchains rather than being explicitly captured in observable artifacts.

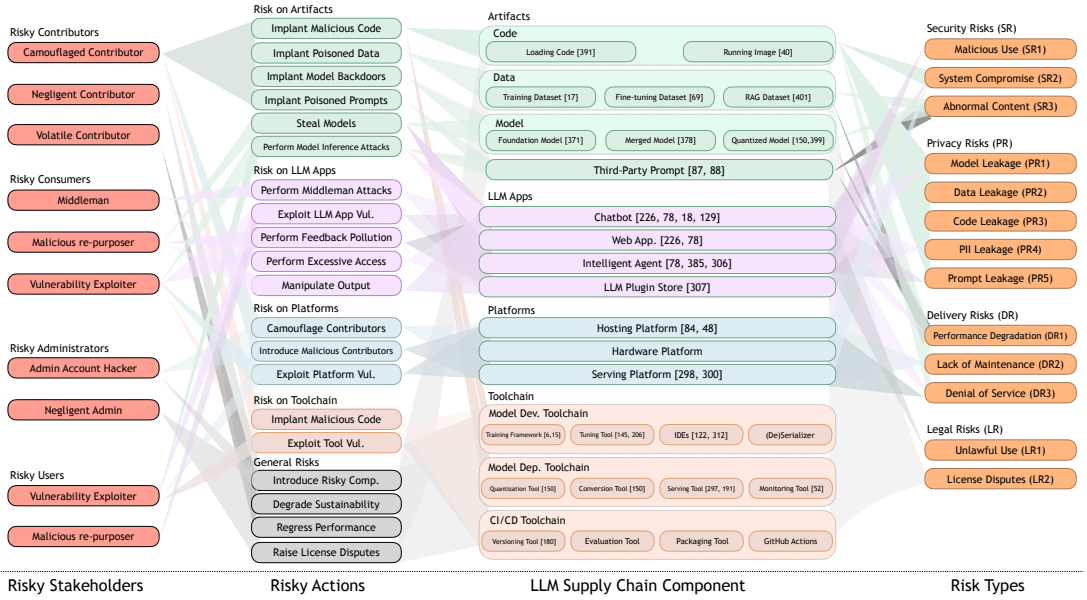

Fig. 4. Risky Actions, Supply Chain Components and Risk Types Introduced by Risky Stakeholders

Summary. We explore both declared and undeclared relations within the LLM supply chain. Undeclared relations, in particular, deserve significant attention due to their inherent complexity in inference and their common occurence across the LLM supply chain. These represent compositional relationships that may introduce new risks and challenges.

5 Risks of the Large Language Model Supply Chain (RQ2)

We introduce the notion of *risky scenarios*. Then, we illustrate risk types, risky stakeholders, risky actions, respectively.

5.1 Risky Scenarios

The risky scenarios can be illustrated that a group of *risky stakeholders* perform some *risky actions* on specific *supply chain components*, causing certain *risk types*. The *risky stakeholders*, *risky actions*, *supply chain components*, and *risk types* are presented from the first column to the fourth column in Figure 4, respectively.

5.2 Risk Types

We categorize four risk types, including security risks, privacy risks, delivery risks and legal risks.

- **Security Risks (SR)** include potential threats such as usage for malicious purposes (SR1), compromising system (SR2), and the emergence of abnormal content (SR3), all of which may raise security concerns for the system.
- **Privacy Risks (PR)** focus on data vulnerabilities, such as model leakage (PR1), data leakage (PR2), personally identifiable information (PII) leakage (PR4), code leakage (PR3), 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), lack of maintenance (DR2), and denial of service (DoS) (DR3) attacks that hinder service availability.

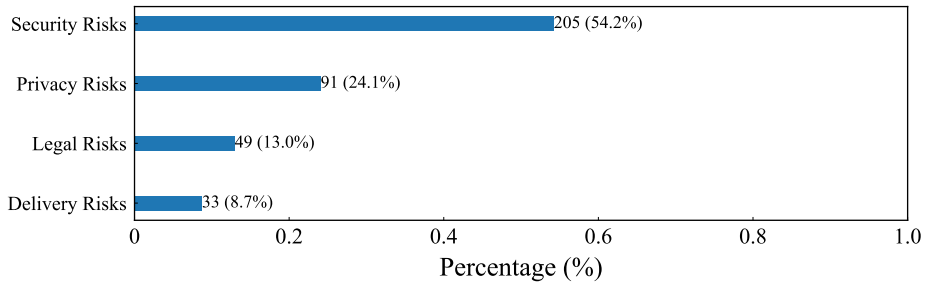

Fig. 5. Distribution of Risk Types that Covered in our Collected Literature

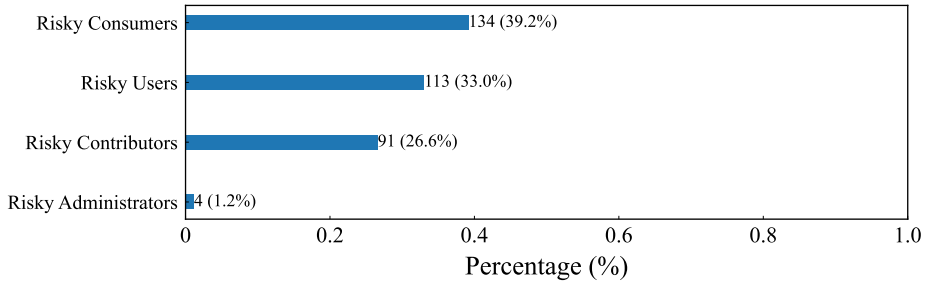

Fig. 6. Distribution of Stakeholders that Covered in our Collected Literature

- **Legal Risks (LR)** emphasize complications arising from copyright disputes, license disputes (LR2), and unlawful use (LR1) of systems, which can lead to legal and compliance challenges.

Figure 5 illustrates the distribution of risk types addressed in the surveyed literature. A majority of the studies (205 papers, 54.2%) concentrate on security risks, followed by privacy risks (91 papers, 24.1%). In contrast, legal risks (49 papers, 13.0%) and delivery risks (33 papers, 8.7%) have received comparatively less attention. This imbalance reflects a research emphasis on technically measurable threats, such as attacks and data leakage, which are more easier for empirical evaluation. However, legal risks (e.g., liability attribution, licensing ambiguity, compliance enforcement) and delivery risks (e.g., deployment misconfiguration, uncontrolled updates) can propagate across multiple stages of the LLM supply chain and affect multiple stakeholders simultaneously. The limited coverage of these risk types suggests that existing mitigation strategies may overlook non-technical yet systemic risks that are critical for the safe and compliant deployment of LLM-based systems.

5.3 Risky Stakeholders

We discuss the risky stakeholders in terms of risky consumers, risky users, risky contributors and risky administrators. We distinguish between risky users and risky consumers: risky users are primarily end users, whereas risky consumers occupy intermediate positions in the supply chain.

- *Risky Contributors:* (a) Malicious contributors can either camouflage legitimate contributors to gain system access or penetrate into systems by leveraging system vulnerabilities. (b) Benign contributors may unintentionally introduce vulnerabilities, either through poor coding practices, lack of maintenance, etc. (c) Volatile contributors may suspend their support under certain circumstances.

- *Risky Consumers:* Attackers can play the role of artifact consumers who can (a) act as middlemen that can inject malicious content during the exchange, or (b) re-develop existing artifact for malicious use. Besides, attackers may also be (c) platform consumers who can exploit platform vulnerabilities.
- *Risky Administrators:* (a) Attackers can hack into an administrator's account, causing severe consequences given the elevated privileges and control. (b) Benign administrators can inadvertently introduce risks into the system through misconfigurations or other unintentional actions.
- *Risky Users:* Users whose purposes are to exploit LLM application vulnerabilities, steal models, conduct jailbreaking or inference attacks, perform excessive access, or pollute the data using user feedbacks.

Figure 6 illustrates the distribution of risky stakeholders addressed that are discussed in the surveyed literature. A majority of the studies (134 papers, 39.2%) concentrate on risky consumers, followed by risky users (113 papers, 33.0%), and risky contributors (91 papers, 26.6%). Few papers discuss the risk introduced by the risky administrators (4 papers, 1.2%). This imbalance indicates a strong research emphasis on externally observable misuse and interaction-level threats, which are relatively easier to study. By comparison, administrator-related risks, such as misconfiguration, insecure deployment, or flawed operational governance, are less visible and often context-dependent, which may explain their limited coverage. However, these risks can have systemic impact, as administrator decisions directly affect model exposure, access control, and downstream safety guarantees. The lack of attention on risky administrators therefore reveals an important gap in existing work.

5.4 Risky Actions

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

5.4.1 Risks on Artifact. The artifact includes code, data, model and prompt. The first four risks involve the implantation of malicious code, poisoned data, model backdoors, and poisoned prompts. These risks can be realized through various implantation mechanisms. We summarize four representative approaches below.

- *Dependency Confusion (Squatting names).* Attackers publish a malicious artifact (i.e., code, data, model or prompt) to a public repository with the same name as a popular private artifact. The artifact manager may unknowingly download the malicious artifact from the public repository instead of the intended private repository [235].
- *Typosquatting.* Attackers upload malicious artifacts with names containing minor typos to imitate popular legitimate artifacts. Accidentally installing those artifacts can lead to security issues and compromise the integrity and security of systems [24].
- *Compromising Legitimate Libraries.* Attackers may target legitimate artifacts already integrated into the LLM supply chain. They can conceal their malicious content within pull requests. In addition, attackers may take control of the artifact from a previous maintainer and release new malicious versions [114].
- *Sharing Cloud Environments.* Attackers can inject malicious artifact into cloud environments and make it publicly available to all cloud users. It was discovered that over a thousand Docker container images were found hiding malicious content [40]. Tomar et al. [304] proposed a threat model that encompasses application-to-container, container-to-container, container-to-host, and host-to-container attack scenarios. They summarize container-layer attack types, including malicious code injection, attacks using cloud container exploitation tools, kernel exploits, tampering, etc.

Implant Malicious Code. Attackers can develop malicious code and attempt to implant it through various methods. For instance, Zhao et al. [391] identified nine malicious dataset-loading scripts on Hugging Face that enable sophisticated attacks, including remote control, browser credential theft, and system reconnaissance. It can cause the risk of malicious use (SR1), system compromise (SR2), and privacy leakage (PR).

Implant Poisoned Data. Attackers can create poisoned data and implant it during critical stages of building the LLM supply chain. For example, Souly et al. [281] found that a near-constant number of poisoned samples (about 250 poisoned documents) can compromise models across a wide range of model sizes (from 600M to 13B parameters) and dataset scales. This finding indicates that poisoned data can be easily injected and become effective during, rendering it an attractive attack vector for adversaries. It can cause the risk of data leakage (PR2) and malicious use (RR1).

- *Training Data Poisoning.* Attackers can manipulate the data for malicious purposes [26, 43, 50, 81, 110, 117, 143, 168, 184, 250, 290, 317, 349, 381], and then upload it to public platforms (*e.g.,* Kaggle) [17, 69, 183]. Once integrated into the LLM supply chain, it would result in the model performance degradation [17, 264, 271, 313, 314].
- *Retrieval Data Poisoning.* Retrieval-augmented generation (RAG) is widely incorporated to provide the LLM with sufficient contextual knowledge. The database (or retrieval sources, such as web content) can also be poisoned, thereby steering the LLM to generate attacker-chosen responses [329, 401].

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. Generally, the models refer to various components, including the foundation model, vocabularies, tokenizers, embedding layers, and auxiliary models, etc [27, 108, 128, 157, 164, 167, 171, 236, 354, 356, 371, 393, 393]. It can cause malicious use (SR1), system compromise (SR2), emergence of abnormal content (SR3), privacy risk (PR), data leakage (PR2), personally identifiable information (PII) leakage (PR4), model leakage (PR1), code leakage (PR3), and prompt leakage (PR5).

- *Component Dependency Attack.* Attackers can manipulate the contents or structures of the composing models of LLMs, affecting the robustness [45, 266] and causing unexpected consequences[113].
- *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 [32, 159, 225].

Implant Poisoned Prompts. Attackers can conceal harmful or misleading prompts into publicly accessible repositories or datasets [105], which are widely shared and reused for downstream users [54, 132, 186]. In fact, several benchmarks have already been proposed for malicious prompt detection [104]. For example, MPDD [127] is a balanced dataset consisting of 39,234 prompts, with half labeled as malicious and the remaining half labeled as benign. It can cause the performance degradation (DR1) of delivery risks.

- *Prompt injection.* Prompt injections disguise malicious instructions as benign, exploiting the incapability of LLMs on distinguishing "good" and "bad" prompts [30, 30, 87, 88, 93, 142, 174, 176, 176, 194, 232, 246, 266, 269, 292, 305, 315, 318, 326, 347, 352, 363, 375, 394].
- *Jailbreaking.* Jailbreaking attempts to bypass safety filters built in the LLMs [22, 36–38, 42, 53, 123, 154, 156, 161, 162, 177, 182, 193, 198, 210, 239, 245, 262, 263, 270, 295, 328, 331, 334, 343, 351, 367, 373, 384, 386, 398], making the LLMs responsive to restricted or insecure content [21, 53, 72, 268, 268, 331, 400].

Beyond the four implantation-related risks discussed above, we also identify inference-stage risks, including model stealing and model inference attacks.

Steal Models. Model stealing attacks are typically carried out by repeatedly querying a target model through its inference interface and collecting the corresponding outputs [12, 18, 73, 96, 192, 227, 260, 280, 309, 332, 360, 368, 390]. By training a surrogate model on these input-output pairs, an attacker can approximate the functionality of the original model without accessing its internal parameters or training data. For instance, the GPT4All project fine-tuned the LLaMA-7B model using one million prompt-response pairs collected via the GPT-3.5-Turbo API, achieving an assistant-style chatbot, highlighting that extraction and can be accomplished within an affordable budget [248]. It can cause model leakage (PR1) of privacy risks.

Perform Model Inference Attacks. Inference attacks aim to exploit LLM models, causing them to inadvertently disclose sensitive information [4, 20, 58, 59, 61, 66, 76, 97, 129, 130, 133, 135, 204, 230, 240, 261, 278, 387, 389, 396]. 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 [224, 286]. Membership inference attacks aim to predict whether a data sample was included in the training data of LLM [188]. For example, in the healthcare domain, Netskope's 2025 report [190] found that sensitive patient data is frequently uploaded to online platforms, raising serious concerns about the use of AI tools such as ChatGPT and Gemini, because these tools are not HIPAA-compliant and may use user-provided data for training. This may lead to the leakage of patients' sensitive information through attribute inference attacks. It can cause data leakage (PR2), personally identifiable information (PII) leakage (PR4), model leakage (PR1), code leakage (PR3), and prompt leakage (PR5).

5.4.2 Risks on LLM Apps. The LLM app can be a web application, an agent or a chatbot.

Perform Middleman Attacks. A Man-in-the-Middle (MitM) attack involves an attacker intercepting communication between two parties without their knowledge [273]. MitM attack in the LLM application occurs 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 [131, 273]. For instance, researchers have found that fake mobile apps impersonating ChatGPT, DALL·E, and other AI platforms are proliferating on alternative app stores, tracking users and stealing sensitive information by abusing trusted branding and permissions [47]. It can cause data leakage (PR2), personally identifiable information (PII) leakage (PR4), model leakage (PR1), code leakage (PR3), and prompt leakage (PR5).

Exploit LLM Application Vulnerabilities. The LLM Plugin Store (*e.g.,* GPT store [217]) and external tools enable users to install plugins to enhance LLM functionalities [19, 56, 93, 249, 340, 345] or enable agentic workflows [141, 166, 175, 200, 303, 337, 358, 370]. However, the external plugins or tool dependencies may embed malicious code or expose exploitable vulnerabilities [172], guiding the model to output incorrect content through prompt injection [310, 338], lead to account takeovers and data leakage [136, 137, 209]. For example, researchers uncovered a critical vulnerability in ChatGPT's plugin that could have allowed attackers to install unauthorized plugins and gain access to users' third-party accounts and sensitive data [255]. It can potentially cause the malicious use (SR1), system compromise (SR2), abnormal content (SR3), data leakage (PR2) and performance degradation (DR1).

Perform Feedback Pollution. LLM applications typically leverage user feedbacks to improve their performance [94, 380] (*e.g.,* reinforcement learning based on human feedback), which can possibly be disrupted by false feedbacks [10, 238, 313]. For example, Wang et al. propose a reward-poisoning attack [321] that manipulates human preference rankings in RLHF to adversarially steer LLM behavior, such as inducing longer generations or embedding trigger-based backdoors. It can potentially cause the malicious use (SR1), system compromise (SR2), emergence of abnormal content (SR3), privacy risk (PR).

Perform Excessive Access. The LLM application may be unresponsive or malfunction due to excessive requests [202, 244, 274, 275]. For example, performing denial of service attacks within the domain of LLMs can significantly increase the energy consumption and latency of the model [272]. In January 2025, DeepSeek experienced a series of intensive DDoS attacks, targeting its API and chat systems around its LLM [70]. It can cause performance degradation (DR1).

Manipulate Output. The responses generated by the LLM application could be manipulated for malicious purposes [1, 99, 131, 155, 173, 201, 222, 285, 302, 308, 355, 361, 364, 376]. For example, cybercriminals could bypass vendor's restrictions to use LLM application for malicious purposes, such as circumventing IP addresses, payment cards and phone numbers controls [25]. It may facilitate malicious use (SR1) and unlawful use (LR1) when manipulated outputs are intentionally exploited.

5.4.3 Risks on Platforms. The platforms support various usages for LLM application such as source code management, data hosting, model training, monitoring, and deployment.

Camouflage Contributors. Attackers can create fake accounts that imitate existing platform accounts (*e.g.*,, official accounts) to publish malicious data, code, models, and more [293]. Users may be misled by these repositories, inadvertently using malicious tools to develop or deploy large models. For example, in the notable xz-utils incident, an attacker gradually contributed benign patches to gain maintainer trust, and then inserted a stealthy backdoor that was propagated to downstream systems [9]. It can cause the risk of malicious use (SR1) and system compromise (SR2).

Introduce Malicious Contributors. Attackers can obtain and exploit credentials of existing accounts or API tokens through various methods [201, 207, 208] (*e.g.*, social engineering). Once in possession of these credentials, attackers can gain unauthorized access to a range of resources and services associated with LLMs [296]. For instance, Hugging Face disclosed that a subset of secrets on its Spaces platform [170], including API tokens, may have been accessed without authorization, underscoring ongoing risks of secret leakage and cross-tenant threats in AI platforms. It can cause the risk of malicious use (SR1) and system compromise (SR2).

Exploit Platform Vulnerabilities. Attackers can identify and take advantage of weaknesses in the underlying platforms. They leverage flaws in cloud services (*e.g.*, SAP AI service [254]) to exploit cross-session information leakage, or retrieve sensitive tokens found in CI/CD artifacts (*e.g.*, GitHub Actions [75]). For example, AWS [299] fixed a critical FlowFixation bug in its Managed Workflows for Apache Airflow service that could have allowed attackers to hijack user sessions and potentially execute remote code or move laterally across services. It can cause the risk of malicious use (SR1), data leakage (PR2) and model leakage (PR1).

5.4.4 Risks on Toolchain. Malicious risks can be introduced at both the development and deployment phases, compromising the integrity and security of the entire pipeline.

Implant Malicious Code. Attackers can compromise the toolchain by injecting malicious code into the toolchain. For instance, "Xcode-Ghost" [333, 383], a malicious version of Xcode, was distributed through unofficial channels. Developers unknowingly used this compromised version to build their apps, resulting in the inclusion of malicious code in the final applications. It can cause the security risk of system compromise (SR2) and the privacy risk (PR).

Exploit Tool vulnerabilities. Attackers exploit known vulnerabilities in development tools, frameworks, or CI/CD components that have not been promptly patched [77]. For instance, the vulnerability of Jenkins Core [121] can lead to remote code execution (RCE) via agent connections. It can cause the security risk of system compromise (SR2), Performance Degradation(DR1) and Denial of Service (DR3) by triggering unexpected errors or system failures.

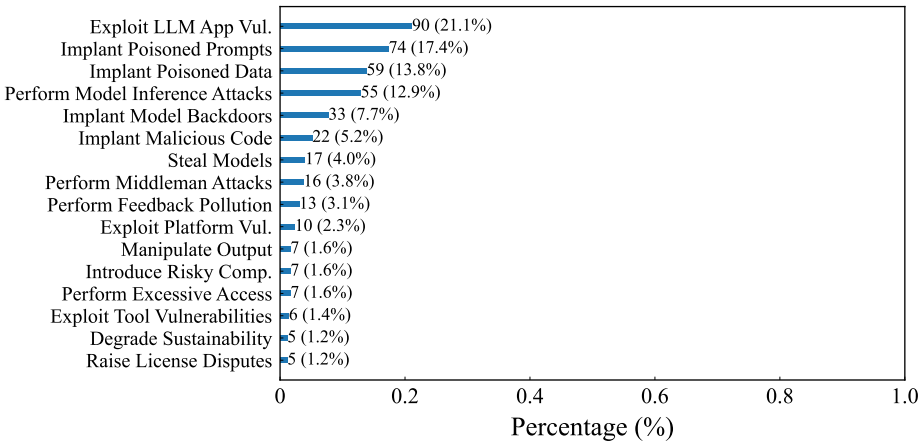

Fig. 7. Distribution of Risky Actions that Covered in our Collected Literature

5.4.5 Generic Risks to LLM Components. Apart from risks that are specific to the four types of components, we elaborate on generic risks to LLM components.

Introduce Risky Components. A naive benign contributor may not be fully aware of the hidden vulnerabilities or security flaws in LLM components. When the benign contributor integrates this component, they inadvertently expose the LLM and its users to potential exploitation. Besides, the benign contributors may submit code that contains security flaws or exploits without malicious intent, ranging from weak encryption practices, insecure data handling, or exploitable logic errors. It can cause the risk of malicious use (SR1), system compromise (SR2), data leakage (PR2), model leakage (PR1), and the emergence of abnormal content (SR3).

Degrade Sustainability. When a contributor stops maintaining a component, updates for security patches, bug fixes, and compatibility improvements are halted. The abandoned component contains outdated and incompatible dependencies that are prone to exploitation or reduce model performance and reliability. For instance, the once-notable deep learning framework Theano [90] ceased active maintenance due to declining community adoption and the rapid evolution of alternative frameworks, resulting in growing compatibility issues and increased security and maintenance risks for downstream users. It can cause the risk of unmaintained (DR2).

Regress Performance. Unintentional actions by contributors or administrators may result in performance regression [28, 124] of the LLM application. For example, a user reported an apparent performance degradation of ChatGPT-4o, noting that the model had become noticeably less effective over time on certain tasks [218]. It can cause the delivery risk of performance degradation (DR1).

Raise License Disputes. While open-source licenses grant general permissions, the owners retain the copyright of the artifact. It introduces potential risks for users who download and use the code, model or data if contributors claim copyrights on their code [148, 181, 288, 301]. In fact, a group of anonymous copyright owners already filed a proposed class-action lawsuit against Microsoft, GitHub, and OpenAI [241], alleging that the companies improperly used open-source code hosted on GitHub to train and monetize AI systems without complying with the applicable open-source licenses. They may enforce restrictions or take legal action if users violate the terms of the license [199]. It can cause the risk of license disputes (LR2).

Figure 7 illustrates the distribution of risky actions addressed in the surveyed literature. The five most frequently discussed actions are exploiting LLM application vulnerabilities (90 papers, 21.1%), implanting poisoned prompts (74

papers, 17.4%), implanting poisoned data (59 papers, 13.8%), performing model inference attacks (55 papers, 12.9%), and implanting model backdoors (33 papers, 7.7%). These actions are largely inherent to the unique characteristics of LLMs and represent threats that are distinct from those observed in traditional software supply chains. In contrast, only a few papers address actions such as degrading sustainability (4 papers, 0.9%), and discontinuation of maintenance (1 paper, 0.2%). This disparity suggests that existing research prioritizes immediate, attack-driven risks over risks that accumulate over time through maintenance neglect, resource exhaustion, or model aging. Given the continuous deployment and iterative evolution of LLM applications, insufficient consideration of such long-term risks may undermine system reliability, security posture, and lifecycle sustainability in practice.

Comparison to traditional software supply chain risks. While many risks in the LLM supply chain stem from traditional software ecosystems—such as *Implant Malicious Code*, *Dependency Confusion*, *Typosquatting*, *Compromising Legitimate Libraries*, *Sharing Cloud Environments*, *Camouflage Contributors*, *Introduce Malicious Contributors*, *Exploit Platform Vulnerabilities*, and *Exploit Tool Vulnerabilities*, LLMs also introduce risks that are specific to their data-driven training paradigm and instruction-following inference behavior. These include *Implant Poisoned Data*, *Implant Model Backdoors*, and *Implant Poisoned Prompts*, as well as inference-stage threats such as *Steal Models* and *Perform Model Inference Attacks*. In addition, LLM applications expose unique interaction-layer risks, including *Prompt Injection*, *Jailbreaking*, *Perform Feedback Pollution*, and *Manipulate Output*, which have no direct counterparts in traditional software systems. By distinguishing between inherited software supply chain risks and LLM-specific risks, this taxonomy highlights the need for specialized governance and mitigation strategies tailored to the unique properties of LLM ecosystems.

Summary. We illustrate the risky scenarios, including high-risk stakeholders, actions, risk types, and their corresponding components within the supply chain. Compared to the traditional software supply chain, the LLM supply chain involves unique risks, such as *implanting poisoned data*, *steal models*, and *perform model inference attacks*. Some risk types are exclusive to LLMs, such as *steal models* and *perform model inference attacks*. The affected components in the LLM supply chain represent a new range of software elements that deserve further investigation and consolidation.

6 Mitigation of Risks in Large Language Model Supply Chain (RQ3)

We elaborate on the mitigation of risks in the LLM supply chain with regard to six aspects, as illustrated in Figure 8

6.1 Proactive Detection

Proactive detection refers to managing systems and processes in a way that anticipates issues, implements preventive measures, and drives improvement rather than merely reacting to problems.

Malware Detection. Malware detection identifies viruses, worms, trojans, and software 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 [311], are effective for identifying binary malware. Recent research has advanced these approaches by modeling the behavior of malicious packages using static characteristics alone [112, 379], as well as integrating both static and dynamic analyses to enhance detection accuracy [60, 106]. This measure can mitigate the risk SR2.

Sanitization. Sanitization involves various techniques to enhance data and prompt security [23, 35, 91, 203, 223, 289, 316, 323, 365, 374], mitigating risks SR3, PR2, and PR4.

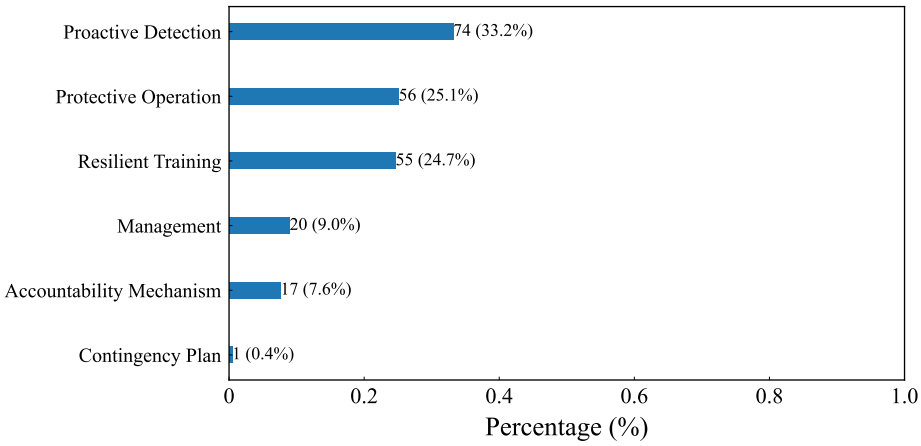

Fig. 8. Distribution of Mitigations that Covered in our Collected Literature

- *Data Filtering.* Data filtering aims to exclude irrelevant, or potentially harmful data. It typically can be realized by replacing sensitive information with non-sensitive tokens. The sensitive information can be identified by named entity recognition algorithms [147]. These methods not only effectively block poisoned or policy-violating content, but also help prevent data leakage by ensuring that shared or processed data cannot be traced back to individuals [41].
- *Maliciousness Classification.* Maliciousness classification employs automated classifiers to detect and label adversarial or malicious inputs, such as poisoned data, harmful prompts, or suspicious queries [41, 55].
- *Data Deduplication.* Jagielski et al. discovered that model privacy attacks are more effective on the training dataset that appears multiple times [118]. Removing duplicate text from the training data [2] helps protect against model memorization of sensitive information.
- *Prompt Sanitization.* Prompt sanitization [265] filters sensitive tokens (e.g., injection tokens [82]), 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. Suo et al. [292] applied signatures to sensitive elements (*e.g.,*'delete file') in the user's prompt and Robey et al. [243] randomly perturb characters in a given input prompt (e.g., inserting random characters), effectively prevent jailbreaking in LLMs. Chen et al .[30] separates prompts and user data into two independent channels to prevent prompt injection.

Anomaly Monitoring. Platforms can apply authorized and restricted access for untrusted accounts. *e.g.,* multi-factor authorization [234]. Apart from that, they can adopt anomaly monitoring by observing developing activities to identify any suspicious behavior [243], such as unauthorized access attempts, changes to account settings, or unexpected pull requests. For example, Danielle et al. [92] leveraged commit messages to detect anomalies and malicious commits. This measure can mitigate the risk SR2.

6.2 Protective Operation

Data Encryption. Data encryption protects data confidentiality by converting it into an unreadable format [276], consisting of three types: symmetric encryption [107], asymmetric encryption [111, 169], and hybrid encryption [3]. With effective key management mechanisms, the data cannot be deciphered, significantly reducing the risk of data privacy breaches [353]. This measure can mitigate the risk PR2 an PR4.

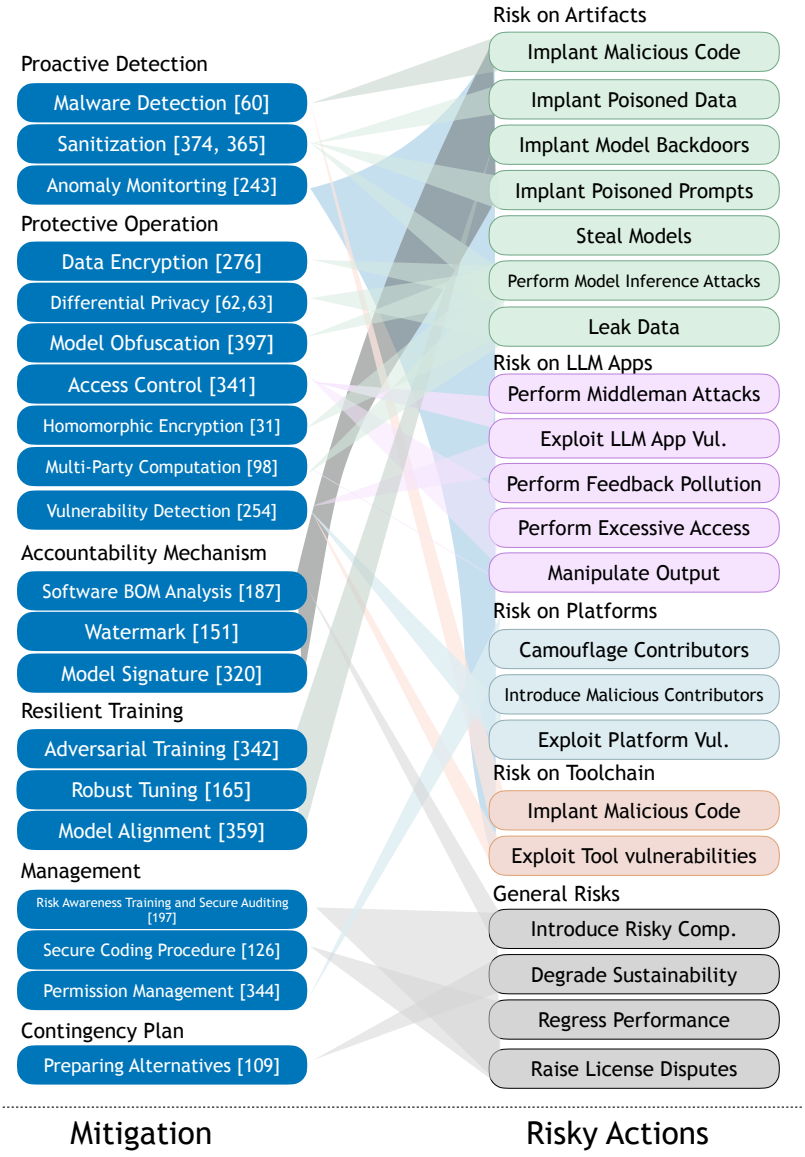

Fig. 9. Mitigations against Risky Actions

Differential Privacy. Differential privacy [8] protects the private information of individual records in a dataset [57, 62, 63, 277, 282]. 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 PR4.

Access Control. Access control mechanisms are essential measures to ensure systems secure interactions and protect sensitive information in systems.

- *Account Access Control.* Authorization (*i.e.,* restricting access to authorized users), authentication (*i.e.,* enforcing mutual authentication and encryption), and rate Limiting (*i.e.,* limiting access to verified users via rate-limiting mechanisms) ensure authorized clients, credible feedback and controlled access rate to prevent system abuse (*e.g.,* Denial of Service (DoS) attacks). This measure can mitigate the risks DR1, DR2 and DR3.
- *Session and Data Isolation.* Session isolation techniques (*e.g.,* sandboxing) separates user sessions, preventing cross-session attacks [341]. Besides, platforms can utilize secrets management tools, log scanning to ensure that sensitive information like tokens and API keys are not exposed in public repositories or logs [75]. This measure can mitigate the risks SR2, PR2, PR4, PR1 and 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-engineering or maliciously exploiting. Zhou et al. [397] proposed an active defense solution for model theft via automated weight obfuscation. Nevertheless, model obfuscation inherently incurs performance degradation [397]. As a result, practitioners must carefully balance the trade-off between intellectual property protection and model efficiency when deploying obfuscation-based defenses. This measure can mitigate the risk PR1.

Homomorphic Encryption. Homomorphic encryption is a form of encryption that allows computations to be performed directly on encrypted data without needing to decrypt it first. Chen et al [31] employs homomorphic encryption to enable privacy-preserving inference for Transformer-based models. This measure can mitigate the risk PR2.

Multi-Party Computation. Li et al. [152] achieve private transformer inference using Secure Multi-Party Computation. Federated learning [98, 346, 392] can be regarded as a particular type of multi-party computation. It enables multiple clients collaboratively train a shared model without transferring their individual data to a central server [158, 388] while keeping the shared model remain secure and resistant to backdoor attacks or malicious data manipulation. This measure can mitigate the risk PR2, PR1 and SR3.

Vulnerability Detection. Vulnerability detection identifies weaknesses or flaws in software or systems that could be exploited by attackers. For example, platforms should also ensure that their 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 [254]. This measure can mitigate the risk SR2.

6.3 Accountability Mechanism

Software Bill-of-Materials Analysis. A Software Bill of Materials (SBOM) is a list of inventory components imported from third-party libraries or data [187], helping users avoid unvetted or potentially risky libraries. Several commercial tools offer SBOM analysis, including Sonatype Management [279], BlackDuck [13], and Microsoft's SBOM tool [195]. Given the widespread use of LLM applications, SBOM analysis can be expanded to encompass not only third-party libraries, but also models, data, prompts, and other related elements. In fact, recent research has already proposed similar notions and specifications, including MLBOM [49, 287], AIBOM [237, 283], and DataBOM [178]. This measure can mitigate the risk SR2 and SR3.

Watermark. Watermark techniques embed identifiable information into data or models to ensure traceability, authenticity, and protection against misuse [151, 228, 348].

- *Data Watermark.* Data watermark protects data privacy by embedding specific watermark identifiers [291, 294] into the training dataset, thus identifying and tracing the data source. This measure can mitigate the risk PR2.
- *Model Watermark.* Model watermark [95, 382] embeds traceable information into the text generated by LLMs to verify the model's source and authenticity without degrading output quality, such as logits modification [134], entropy-based embedding [149], multi-bit payload [322], and token sampling [138]. This measure can mitigate the risk PR1.

Model Signature. Model signature helps to track and manage the origin of models [320]. Jiang et al. [125] 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 PR1.

6.4 Resilient Training

Model Alignment. Model alignment [11, 16, 231, 233, 252, 319, 324, 330, 350, 357, 359, 377] ensures that a model's outputs and behaviors align with human values and expectations. Reinforcement Learning from Human Feedback [160, 189, 219] 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 designed to enhance model robustness by exposing the model to adversarial examples [29, 39, 80, 100, 101, 115, 119, 139, 220, 251, 327, 336, 342, 366, 369] 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.

Robust Tuning. Robust tuning enhances the defense of LLMs against prompt risks by incorporating specific techniques during tuning [5, 33, 83, 165]. For example, Yue et al. [372] fine-tuning a generative language model with differential privacy to generate privacy-preserving synthetic text. Ozdayi et al. [221] leverage prompt-tuning to control the extraction rates of memorized content, avoiding attack modifying LLM weights. This measure can mitigate the risk SR3.

6.5 Management

Risk Awareness Training and Secure Auditing. Administrators should be vigilant when components are incorporated into their software systems [140, 197, 259]. 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. Furthermore, they should conduct regular security audits to assess LLM supply chain component security and compliance. This measure can mitigate the risks SR2 and LR2.

Secure Coding Procedure. Adhering to secure coding practices is fundamental [126] 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.

Permission Management. Permission management includes enforcing strict vetting processes to ensure that only trusted components are introduced into the system [344]. Additionally, employing role-based access control (RBAC)

limits permissions, ensuring that users and LLM supply chain components have only the necessary access to perform their functions, thereby reducing the attack surface. This measure can mitigate the risks SR1, SR3 and LR1.

6.6 Contingency Plan

Preparing Alternatives. The contingency plan defines a set of predefined actions and response procedures when unexpected events occur. Preparing alternatives involves developing backup plans to mitigate risks arising from reliance on components that may become unsupported or discontinued [109]. Moreover, advanced replacement techniques are also needed to help migrate the original component into a new one with the least effort and cost. For example, Almeida et al. propose an LLM-based approach for automated SQLAlchemy library migration [7]. This measure can mitigate the risks SR2, DR2 and DR3.

Figure 8 presents the distribution of mitigation strategies discussed in the surveyed literature. The majority of studies propose solutions based on proactive detection (74 papers, 33.2%) and protective operation (56 papers, 25.1%). Resilient training techniques also account for a significant portion, reflecting efforts to enhance model robustness during development. The fourth and fifth most discussed strategies involve *management* (20 papers, 9.0%) practices and *accountability mechanisms* (17, 7.6%), which primarily address socio-organizational aspects of risk mitigation. Notably, only one paper discusses the use of contingency planning as a mitigation approach.

Summary. We discusses various strategies to mitigate risks in the LLM supply chain. Key measures include malware detection and Software Bill-of-Materials (SBOM) analysis to address risks from third-party libraries, and privacy-preserving techniques such as data deduplication, encryption, and differential privacy to secure datasets. These strategies address a comprehensive range of risks, ensuring a robust framework for LLM development, deployment, and operation.

7 Evidence from Existing Systems

We examine real-world security incidents in the LLM supply chain and to validate the applicability of our risk taxonomy and mitigation strategies. We selected four cases, which were purposefully selected based on: real-world impact, coverage of LLM supply chain components, and diversity of LLM supply chain risks and mitigation. For each case, we conducted a targeted Google search using case-specific keywords and reviewed the first ten results on the first page. We examined the associated documents to understand how the reported incidents illustrate concrete risk scenarios and how these scenarios map to our taxonomy. To reduce misunderstanding and misinterpretation, one author first reviewed the selected sources and summarized the identified risk scenarios and corresponding mitigation measures. A second author then independently cross-checked the summary. Any discrepancies were discussed and resolved through consensus. Table 2 presents the analyzed systems and their mapped compositions, risks, and mitigations.

Probllama. Ollama is an open-source and widely-used framework that allows users to operate LLMs. Recently, researchers uncovered 6 vulnerabilities in Ollama [214]. The vulnerabilities could allow an attacker to carry out a wide-range of malicious actions, including Denial of Service (DoS) attacks, model poisoning, model stealing, etc. Researchers highlighted that the inherent lack of authentication support in these tools makes them vulnerable when exposed to external environments. Specifically, the *CVE-2024-37032*, known as the "Probllama" vulnerability, allows arbitrary file writes through path traversal, which could lead to remote code execution, data breaches, or corruption of critical files in environments with elevated privileges (e.g., Docker). The *risk scenario* is that *Vulnerability Exploiter* performs *Stealing Models* and *Performing Excessive Access* on AI models, causing *Model Leakage (PR1)*, and *Denial of Service (DR3)*, which

Table 2. Mapping between Systems and Composition, Risks and Mitigation

System	Composition	Risks			Mitigation
		Risky Stakeholder	Risky Actions	Risk Types	
Ollama	Artifact	Risky Consumer	*Stealing Models*, *Performing Excessive Access*	*Model Leakage (PR1)*, *Denial of Service (DR3)*	*Access Control*, *Permission Management*
Ray	Platform	Risky User	*Exploit Platform Vul.*, *Stealing Models*	*Data Leakage (PR2)*, *Model Leakage (PR1)*, *PII Leakage (PR4)*	*Vulnerability Detection*, *Access Control*
"nullifAI" and HF Safetensors	Toolchain	Risky Consumer	*Exploiting LLM App. Vul.*, *Exploiting Platform Vul.*	*System Compromise (SR2)*, *Privacy Risks (PR)*	*Vulnerability Detection* and *Malware Detection*
DeepSeek	Applications	Risky User, Risky Consumer	*Performing Middleman Attacks*, *Exploiting LLM App. Vul.*, *Introducing Risky Components*	*System Compromise (SR2)*, *Privacy Risks (PR)*	*Risk Awareness Training and Secure Auditing*, *Software BOM Analysis*, *Malware Detection*

includes *Chatbots*, *Web Apps* and *Intelligent Agents*, etc. The corresponding mitigation measures include *Access Control* and *Permission Management* [214].

Ray. Ray is a distributed computing framework designed to simplify the process of scaling AI applications. It was reported that attackers have been exploiting a missing authentication vulnerability in the Ray AI framework to compromise hundreds of clusters [215]. By default, Ray does not enforce authentication and does not support any type of authorization methods. Reporters say that hundreds of Ray clusters were hacked via this bug [257], with the attackers stealing AI models and data, database credentials, password hashes, SSH keys, and OpenAI, HuggingFace, and Stripe tokens. The *risk scenario* is that *Vulnerability Exploiter* performs *Stealing Artifacts* including *data*, *model* and *code*, etc, causing *Data Leakage (PR2)*, *Model Leakage (PR1)*, *PII Leakage (PR4)*. The corresponding mitigation measures include *Vulnerability Detection* and *Access Control* .

"nullifAI" and Hugging Face Safetensors. Pickle is a popular Python module for serializing and deserializing ML models. However, Pickle is considered an unsafe data format that allows Python code to be executed during deserialization, which can lead to arbitrary code execution. A research team came upon two Hugging Face models containing malicious code exploiting the unsafe mechanisms of Pickle, named as "nullifAI" [242]. The *risk scenario* is that *Vulnerability Exploiter* performs *Exploiting Artifact Vulnerabilities* including *data*, *model* and *code*, etc, causing *System Compromise (SR2)* and *Privacy Risks (PR)*.

Furthermore, the Hugging Face introduces a new safe serialization format called Safetensors [102] to mitigate the supply chain risk posed by vulnerable serialization formats. They created a conversion service to convert any PyTorch model contained within a repository into a Safetensors alternative via a pull request. However, researchers demonstrate that attackers could compromise the Safetensors conversion space and its associated service bot [102]. The *risk scenario* is that *Vulnerability Exploiter* performs *Exploiting Platform Vulnerabilities*, causing *System Compromise (SR2)* and *Privacy Risks (PR)*. The corresponding mitigation measures include *Vulnerability Detection* and *Malware Detection*.

DeepSeek Brand Impersonation. DeepSeek AI chatbot quickly gained international attention, making it a prime target for abuse. Attackers leverages a tactic known as brand impersonation, creating look-alike websites to deceive users and steal private information [256, 402]. They also uses a fake CAPTCHA page to deceive users into executing a malicious command. The *risk scenario* is that *System Compromise (SR2)* and *Privacy Risks (PR)* are caused by *Vulnerability Exploiter Performing Middleman Attacks* and *Exploiting LLM App. Vulnerabilities*, and *Negligent User Introducing Risky*

Components. The corresponding mitigation measures include *Risk Awareness Training and Secure Auditing*, *Software BOM Analysis* and *Malware Detection*.

8 Discussion

We discuss the key challenges in mitigating risks within the LLM supply chain.

Opaque Dependencies. The first challenge lies in constructing a precise and comprehensive Bill of Materials (BOM) for LLMs. In traditional software, an SBOM enumerates source code files, libraries, dependencies, and versions. By contrast, many LLM components are inherently opaque: datasets are massive and often untraceable (see *Training Relations from Training and Fine-tuning Dataset* in Sec. 4.3.2), pretrained weights embed external knowledge without clear provenance (see *Artifact Cloning Relations* in Sec. 4.3.2), and prompts are transient (see *Prompt Augmentation Relations from RAG Dataset and Third-party Prompts* in Sec. 4.3.2) yet capable of introducing risks. Thus, recovering the opaque dependencies around the LLM (BOM) is fundamentally more difficult, as it must capture not only static dependencies but also probabilistic factors that contribute to model behavior. The absence of standardized metadata formats for LLM artifacts further hinders reproducibility, auditability, and transparency.

Unquantifiable Risks and Undetermined Risk Propagation. The second challenge lies in the fact that many risks in the LLM supply chain are difficult to quantify and their propagation remains poorly understood. Unlike traditional software vulnerabilities, which can often be measured in terms of severity and exploitability, LLM-specific risks such as poisoned data, poisoned prompts, feedback pollution, or model stealing (see Sec. 5.4) lack clear metrics for evaluation. For instance, the impact of poisoned samples on model integrity (see Sec. 5.4) is primarily supported by empirical evidence rather than formal guarantees, leaving the propagation and extent of such risks insufficiently characterized. Likewise, while prompt injection attacks (see Sec. 5.4) show that adversaries can steer model outputs via crafted inputs, their probability of success and downstream consequences are still poorly understood. Moreover, once such risks are introduced, their trajectories are uncertain: Do poisoned weights continue to exert influence after fine-tuning or prompt sanitization? Do vulnerabilities persist when models are reused, distilled, or integrated into downstream systems? These unquantifiable risks and uncertain propagation patterns complicate systematic analysis, making exposure difficult to assess and undermining the long-term trustworthiness of the supply chain.

Imperfect Risks Mitigation. Software vulnerabilities can often be patched or dependencies replaced in traditional software, but risks embedded within an LLM are persistent and rarely removable. Once a model is trained on poisoned, biased, copyrighted, or sensitive data, it cannot simply be "uninstalled" or "patched". For example, one of the sanitization technique leverages maliciousness classification (see Sec. 6.4), which inherently cannot eliminate risks entirely but instead aim to contain them at a manageable level. Retraining from scratch is prohibitively costly, while fine-tuning, filtering, or post-hoc defenses typically address only part of the problem and may even introduce new side effects. For instance, resilient training (see Sec. 6.4) can improve model robustness and reduce susceptibility to malicious inputs or harmful behaviors. However, these approaches require substantial computational and human resources, which may not generalize to unseen attacks, and only partially mitigate the embedded risks. Consequently, mitigation remains imperfect, offering only partial containment rather than complete eradication of risks. This underscores the need for more robust strategies, such as auditable training pipelines, verifiable provenance tracking, and continual monitoring of model behavior, while acknowledging that such measures still cannot guarantee absolute trustworthiness.

Addressing these challenges yields several important benefits. First, an accurate LLM BOM provides a foundation for proactively defending against risks such as *implanting poisoned data*, *implanting malicious code*, *implanting poisoned prompts*, *performing feedback pollution*, or *raising licensing disputes*. Second, it enables auditable and transparent

development practices, which are essential for the trustworthy deployment of LLMs in high-stakes domains. Third, it enhances traceability across the supply chain, facilitating error diagnosis, model version control, and the safeguarding of sensitive information. Collectively, these benefits strengthen the reliability, accountability, and long-term sustainability of the LLM ecosystem.

9 Conclusions

The rapid growth of LLMs 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.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (Grant No. 62332005, 62372114 and 62402342), and also supported by Shanghai Sailing Program (No. 24YF2749500).

References

[1] [n. d.]. Teach a GPT to Phish.
[2] 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).
[3] 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).
[4] Divyansh Agarwal, Alexander R Fabbri, Philippe Laban, Shafiq Joty, Caiming Xiong, and Chien-Sheng Wu. 2024. Investigating the prompt leakage effect and black-box defenses for multi-turn llm interactions. *arXiv preprint arXiv:2404.16251* (2024).
[5] Udit Kumar Agarwal, Abraham Chan, and Karthik Pattabiraman. 2023. Resilience Assessment of Large Language Models under Transient Hardware Faults. *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)* (2023), 659–670.
[6] Facebook AI. 2024. PyTorch. Retrieved May 30, 2024 from https://pytorch.org
[7] Aylton Almeida, Laerte Xavier, and Marco Túlio Valente. 2024. Automatic Library Migration Using Large Language Models: First Results. *Proceedings of the 18th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* (2024).
[8] 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.
[9] 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
[10] 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).
[11] Rishabh Bhardwaj and Soujanya Poria. 2023. Red-Teaming Large Language Models using Chain of Utterances for Safety-Alignment. *ArXiv* abs/2308.09662 (2023).
[12] Lewis Birch, William Hackett, Stefan Trawicki, Neeraj Suri, and Peter Garraghan. 2023. Model Leeching: An Extraction Attack Targeting LLMs. *ArXiv* abs/2309.10544 (2023).
[13] BlackDuck. 2002. BlackDuck. Retrieved May 30, 2024 from https://www.blackduck.com/software-composition-analysis-tools/black-duck-sca.html
[14] 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/
[15] Google Brain. 2024. TensorFlow. Retrieved May 30, 2024 from https://www.tensorflow.org
[16] Bochuan Cao, Yu Cao, Lu Lin, and Jinghui Chen. 2023. Defending Against Alignment-Breaking Attacks via Robustly Aligned LLM. In *Annual Meeting of the Association for Computational Linguistics*.
[17] Nicholas Carlini, Matthew Jagielski, Christopher A Choquette-Choo, Daniel Paleka, Will Pearce, Hyrum Anderson, Andreas Terzis, Kurt Thomas, and Florian Tramèr. 2024. Poisoning web-scale training datasets is practical. In *2024 IEEE Symposium on Security and Privacy (SP)*. IEEE, 407–425.

[18] 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).

[19] Chun Fai Chan, Daniel Wankit Yip, and Aysan Esmradi. 2023. Detection and Defense Against Prominent Attacks on Preconditioned LLM-Integrated Virtual Assistants. *2023 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)* (2023), 1–5.

[20] Nishanth Chandran, Sunayana Sitaram, Divya Gupta, Rahul Sharma, Kashish Mittal, and Manohar Swaminathan. 2024. Private Benchmarking to Prevent Contamination and Improve Comparative Evaluation of LLMs. *ArXiv* abs/2403.00393 (2024).

[21] Zhiyuan Chang, Mingyang Li, Yi Liu, Junjie Wang, Qing Wang, and Yang Liu. 2024. Play Guessing Game with LLM: Indirect Jailbreak Attack with Implicit Clues. In *Annual Meeting of the Association for Computational Linguistics*.

[22] Patrick Chao, Alexander Robey, Edgar Dobriban, Hamed Hassani, George J. Pappas, and Eric Wong. 2023. Jailbreaking Black Box Large Language Models in Twenty Queries. *2025 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)* (2023), 23–42.

[23] Yiannis Charalambous, Norbert Tihanyi, Ridhi Jain, Youcheng Sun, Mohamed Amine Ferrag, and Lucas C. Cordeiro. 2023. A New Era in Software Security: Towards Self-Healing Software via Large Language Models and Formal Verification. *2025 IEEE/ACM International Conference on Automation of Software Test (AST)* (2023), 136–147.

[24] 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/

[25] checkpoint. 2024. *Russian Hackers Attempt to Bypass OpenAI's Restrictions for Malicious Use of ChatGPT.* Retrieved April 20, 2024 from https://blog.checkpoint.com/2023/01/13/russian-hackers-attempt-to-bypass-openais-restrictions-for-malicious-use-of-chatgpt/

[26] Jian Chen, Xuxin Zhang, Rui Zhang, Chen Wang, and Ling Liu. 2021. De-Pois: An Attack-Agnostic Defense against Data Poisoning Attacks. *IEEE Transactions on Information Forensics and Security* 16 (2021), 3412–3425.

[27] 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).

[28] Lingjiao Chen, Matei Zaharia, and James Y. Zou. 2023. How is ChatGPT's behavior changing over time? *ArXiv* abs/2307.09009 (2023).

[29] Musheng Chen, Guowei He, and Junhua Wu. 2024. ZDDR: A Zero-Shot Defender for Adversarial Samples Detection and Restoration. *IEEE Access* 12 (2024), 39081–39094.

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

[31] 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).

[32] 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.

[33] Xiaoyi Chen, Siyuan Tang, Rui Zhu, Shijun Yan, Lei Jin, Zihao Wang, Liya Su, Zhikun Zhang, Xiaofeng Wang, and Haixu Tang. 2023. The Janus Interface: How Fine-Tuning in Large Language Models Amplifies the Privacy Risks. *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security* (2023).

[34] Ziqian Chen, Zekai Chen, Susheng Wu, Bihuan Chen Chen, Wenyan Song, Yiheng Huang, Zhuotong Zhou, Yiheng Cao, and Xin Peng. 2026. A First Look at Model Supply Chain: From the Risk Perspective. In *Proceedings of the 48th International Conference on Software Engineering*.

[35] Zhiyuan Chen, Yu Li, Suochao Zhang, Jingbo Zhou, Jiwen Zhou, Chenfu Bao, and Dianhai Yu. 2024. A Framework for Cost-Effective and Self-Adaptive LLM Shaking and Recovery Mechanism. *ArXiv* abs/2403.07283 (2024).

[36] Zhaorun Chen, Zhuokai Zhao, Wenjie Qu, Zichen Wen, Zhiguang Han, Zhihong Zhu, Jiaheng Zhang, and Huaxiu Yao. 2024. Pandora: Detailed llm jailbreaking via collaborated phishing agents with decomposed reasoning. In *ICLR 2024 Workshop on Secure and Trustworthy Large Language Models*.

[37] Yixin Cheng, Markos Georgopoulos, Volkan Cevher, and Grigorios Chrysos. 2024. Leveraging context in jailbreaking attacks. In *ICLR 2024 Workshop on Secure and Trustworthy Large Language Models*.

[38] Yixin Cheng, Markos Georgopoulos, Volkan Cevher, and Grigorios G. Chrysos. 2024. Leveraging the Context through Multi-Round Interactions for Jailbreaking Attacks. *ArXiv* abs/2402.09177 (2024).

[39] Steffi Chern, Zhen Fan, and Andy Liu. 2024. Combating Adversarial Attacks with Multi-Agent Debate. *ArXiv* abs/2401.05998 (2024).

[40] 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/

[41] Chun Jie Chong, Chenxi Hou, Zhihao Yao, and Seyed Mohammadjavad Seyed Talebi. 2024. Casper: Prompt Sanitization for Protecting User Privacy in Web-Based Large Language Models. *2025 IEEE 12th International Conference on Cyber Security and Cloud Computing (CSCloud)* (2024), 122–133.

[42] Junjie Chu, Yugeng Liu, Ziqing Yang, Xinyue Shen, Michael Backes, and Yang Zhang. 2024. Comprehensive Assessment of Jailbreak Attacks Against LLMs. *ArXiv* abs/2402.05668 (2024).

[43] Antonio Emanuele Cinà, Kathrin Grosse, Ambra Demontis, Battista Biggio, Fabio Roli, and Marcello Pelillo. 2022. Machine Learning Security Against Data Poisoning: Are We There Yet? *Computer* 57 (2022), 26–34.

[44] 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).

[45] 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.

[46] Xinyue Cui, Johnny Wei, Swabha Swayamdipta, and Robin Jia. 2025. Robust data watermarking in language models by injecting fictitious knowledge. In *Findings of the Association for Computational Linguistics: ACL 2025*. 14292–14306.

[47] cyberpress. 2025. Malicious ChatGPT Apps Are Tracking Users and Stealing Sensitive Information. Retrieved January 1, 2026 from https://cyberpress.org/malicious-chatgpt-apps/

[48] cybersecuritydive. 2024. *GitHub vulnerability raises risk of open source supply chain attack.* Retrieved April 20, 2024 from https://www.cybersecuritydive.com/news/github-vulnerability-supply-chain-attack/635127/

[49] cyclonedx. 2025. *CyClone MLBOM.* Retrieved December 20, 2025 from https://cyclonedx.org/capabilities/mlbom/

[50] Avisha Das, Amara Tariq, Felipe Batalini, Boddhisattwa Dhara, and Imon Banerjee. 2024. Exposing Vulnerabilities in Clinical LLMs Through Data Poisoning Attacks: Case Study in Breast Cancer. *medRxiv* (2024).

[51] Badhan Chandra Das, M Hadi Amini, and Yanzhao Wu. 2024. Security and privacy challenges of large language models: A survey. *arXiv preprint arXiv:2402.00888* (2024).

[52] datadoghq. 2024. *What Is LLM Observability and How Does it Work?* Retrieved April 20, 2024 from https://www.datadoghq.com/knowledge-center/llm-observability/

[53] 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 *Proceedings 2024 Network and Distributed System Security Symposium*.

[54] Yimo Deng and Huangxun Chen. 2023. Divide-and-Conquer Attack: Harnessing the Power of LLM to Bypass the Censorship of Text-to-Image Generation Model. *ArXiv* abs/2312.07130 (2023).

[55] Himani Deshpande, Dhawal Chaudhari, Tanmay Sarode, and Ayush Kamath. 2024. Exploring LLM and Neural Networks Towards Malicious Prompt Detection. *2024 5th International Conference on Electronics and Sustainable Communication Systems (ICESC)* (2024), 1531–1535.

[56] Tian Dong, Guoxing Chen, Shaofeng Li, Minhui Xue, Rayne Holland, Yan Meng, Zhen Liu, and Haojin Zhu. 2023. Unleashing Cheapfakes through Trojan Plugins of Large Language Models. *ArXiv* abs/2312.00374 (2023).

[57] Haonan Duan, Adam Dziedzic, Nicolas Papernot, and Franziska Boenisch. 2023. Flocks of Stochastic Parrots: Differentially Private Prompt Learning for Large Language Models. *ArXiv* abs/2305.15594 (2023).

[58] Haonan Duan, Adam Dziedzic, Mohammad Yaghini, Nicolas Papernot, and Franziska Boenisch. 2024. On the Privacy Risk of In-context Learning. *ArXiv* abs/2411.10512 (2024).

[59] Michael Duan, Anshuman Suri, Niloofar Mireshghallah, Sewon Min, Weijia Shi, Luke Zettlemoyer, Yulia Tsvetkov, Yejin Choi, David Evans, and Hannaneh Hajishirzi. 2024. Do membership inference attacks work on large language models? *arXiv preprint arXiv:2402.07841* (2024).

[60] 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*.

[61] Jan Dubi'nski, A. D. Kowalczuk, Stanislaw Pawlak, Przemyslaw Rokita, Tomasz Trzci'nski, and Paweł Morawiecki. 2023. Towards More Realistic Membership Inference Attacks on Large Diffusion Models. *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)* (2023), 4848–4857.

[62] 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.

[63] 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.

[64] 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.

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

[66] Jonathan Evertz, Merlin Chlosta, Lea Schönherr, and Thorsten Eisenhofer. 2024. Whispers in the Machine: Confidentiality in LLM-integrated Systems. *ArXiv* abs/2402.06922 (2024).

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

[68] 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).

[69] far.ai. 2024. *GPT-4o Guardrails Gone: Data Poisoning and Jailbreak-Tuning.* Retrieved April 20, 2024 from https://far.ai/post/2024-10-poisoning/#1-malicious-fine-tuning

[70] fastnetmon. 2025. DeepSeek DDoS Attacks Explained – what really happened? Retrieved January 1, 2026 from https://fastnetmon.com/2025/02/05/deepseek-ddos-attacks-explained-what-really-happened/

[71] Qizhang Feng, Siva Rajesh Kasa, Hyokun Yun, Choon Hui Teo, and Sravan Babu Bodapati. 2024. Exposing privacy gaps: Membership inference attack on preference data for llm alignment. *arXiv preprint arXiv:2407.06443* (2024).

[72] Yingchaojie Feng, Zhizhang Chen, Zhining Kang, Sijia Wang, Haoyu Tian, Wei Zhang, Minfeng Zhu, and Wei Chen. 2024. JailbreakLens: Visual Analysis of Jailbreak Attacks Against Large Language Models. *IEEE Transactions on Visualization and Computer Graphics* 31 (2024), 8668–8682.

[73] Matthew Finlayson, Xiang Ren, and Swabha Swayamdipta. 2024. Logits of API-Protected LLMs Leak Proprietary Information. *ArXiv* abs/2403.09539 (2024).

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

[75] 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

[76] Wenjie Fu, Huandong Wang, Chen Gao, Guanghua Liu, Yong Li, and Tao Jiang. 2023. Practical Membership Inference Attacks against Fine-tuned Large Language Models via Self-prompt Calibration. *ArXiv* abs/2311.06062 (2023).

[77] Xiaohan Fu, Zihan Wang, Shuheng Li, Rajesh K. Gupta, Niloofar Mireshghallah, Taylor Berg-Kirkpatrick, and Earlence Fernandes. 2023. Misusing Tools in Large Language Models With Visual Adversarial Examples. *ArXiv* abs/2310.03185 (2023).

[78] Kuofeng Gao, Tianyu Pang, Chao Du, Yong Yang, Shu-Tao Xia, and Min Lin. 2024. Denial-of-service poisoning attacks against large language models. *arXiv preprint arXiv:2410.10760* (2024).

[79] Vahid Garousi and Mika V Mäntylä. 2016. A systematic literature review of literature reviews in software testing. *Information and Software Technology* 80 (2016), 195–216.

[80] Suyu Ge, Chunting Zhou, Rui Hou, Madian Khabsa, Yi-Chia Wang, Qifan Wang, Jiawei Han, and Yuning Mao. 2023. MART: Improving LLM Safety with Multi-round Automatic Red-Teaming. In *North American Chapter of the Association for Computational Linguistics*.

[81] Jonas Geiping, Liam H. Fowl, W. Ronny Huang, Wojciech Czaja, Gavin Taylor, Michael Moeller, and Tom Goldstein. 2020. Witches' Brew: Industrial Scale Data Poisoning via Gradient Matching. *ArXiv* abs/2009.02276 (2020).

[82] Runpeng Geng, Yanting Wang, Chenlong Yin, Minhao Cheng, Ying Chen, and Jinyuan Jia. 2025. PISanitizer: Preventing Prompt Injection to Long-Context LLMs via Prompt Sanitization.

[83] Shaona Ghosh, Prasoon Varshney, Erick Galinkin, and Christopher Parisien. 2024. AEGIS: Online Adaptive AI Content Safety Moderation with Ensemble of LLM Experts. *ArXiv* abs/2404.05993 (2024).

[84] gitguardian. 2024. *researcher finds github admin credentials thanks to misconfigurations.* Retrieved April 20, 2024 from https://blog.gitguardian.com/researcher-finds-github-admin-credentials-thanks-to-misconfigurations/

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

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

[87] GitHub. 2024. *llm security prompt injection.* Retrieved April 20, 2024 from https://github.com/sinanw/llm-security-prompt-injection

[88] GitHub. 2024. *malicious prompts.* Retrieved April 20, 2024 from https://github.com/llm-security-research/malicious-prompts

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

[90] github. 2025. Theano. Retrieved January 1, 2026 from https://github.com/Theano/Theano?tab=readme-ov-file

[91] David Glukhov, Ilia Shumailov, Yarin Gal, Nicolas Papernot, and Vardan Papyan. 2023. LLM Censorship: A Machine Learning Challenge or a Computer Security Problem? *ArXiv* abs/2307.10719 (2023).

[92] 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*.

[93] 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.

[94] 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).

[95] Batu Guan, Yao Wan, Zhangqian Bi, Zheng Wang, Hongyu Zhang, Pan Zhou, and Lichao Sun. 2024. CodeIP: A Grammar-Guided Multi-Bit Watermark for Large Language Models of Code. *ArXiv* abs/2404.15639 (2024).

[96] 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.

[97] Christian Gustavsson. 2024. Approximation-based monitoring of ongoing model extraction attacks: model similarity tracking to assess the progress of an adversary.

[98] Shanshan Han, Baturalp Buyukates, Zijian Hu, Han Jin, Weizhao Jin, Lichao Sun, Xiaoyang Wang, Wenxuan Wu, Chulin Xie, Yuhang Yao, Kai Zhang, Qifan Zhang, Yuhui Zhang, Salman Avestimehr, and Chaoyang He. 2023. FedSecurity: A Benchmark for Attacks and Defenses in Federated Learning and Federated LLMs. *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (2023).

[99] Tianyu Han, Sven Nebelung, Firas Khader, Tian Wang, Gustav Mueller-Franzes, Christiane Kuhl, Sebastian Forsch, Jens Kleesiek, Christoph Haarburger, Keno Kyrill Bressem, Jakob Nikolas Kather, and Daniel Truhn. 2023. Medical Foundation Models are Susceptible to Targeted Misinformation Attacks. *ArXiv* abs/2309.17007 (2023).

[100] Jingxuan He and Martin T. Vechev. 2023. Large Language Models for Code: Security Hardening and Adversarial Testing. *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security* (2023).

[101] Alec Helbling, Mansi Phute, Matthew Hull, Sebastian Szyller, and Duen Horng Chau. 2023. LLM Self Defense: By Self Examination, LLMs Know They Are Being Tricked. *ArXiv* abs/2308.07308 (2023).

[102] hiddenlayer. 2024. *HIJACKING SAFETENSORS CONVERSION ON HUGGING FACE.* Retrieved April 20, 2024 from https://hiddenlayer.com/innovation-hub/silent-sabotage/

[103] hiddenlayer. 2024. *SHADOWLOGIC.* Retrieved April 20, 2024 from https://hiddenlayer.com/innovation-hub/shadowlogic/

[104] hiddenlayer. 2025. Evaluating Prompt Injection Datasets. Retrieved January 1, 2026 from https://hiddenlayer.com/innovation-hub/evaluating-prompt-injection-datasets/

[105] Keegan Hines, Gary Lopez, Matthew Hall, Federico Zarfati, Yonatan Zunger, and Emre Kiciman. 2024. Defending Against Indirect Prompt Injection Attacks With Spotlighting. *ArXiv* abs/2403.14720 (2024).

[106] 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.*

[107] 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.

[108] Hai Huang, Zhengyu Zhao, Michael Backes, Yun Shen, and Yang Zhang. 2023. Composite Backdoor Attacks Against Large Language Models. In *NAACL-HLT.*

[109] Ken Huang, Jerry Huang, and Daniele Catteddu. 2024. GenAI data security. In *Generative AI security: Theories and practices.* Springer, 133–162.

[110] W. Ronny Huang, Jonas Geiping, Liam H. Fowl, Gavin Taylor, and Tom Goldstein. 2020. MetaPoison: Practical General-purpose Clean-label Data Poisoning. *ArXiv* abs/2004.00225 (2020).

[111] 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.

[112] 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.*

[113] 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.

[114] 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

[115] Nanna Inie, Jonathan Stray, and Leon Derczynski. 2023. Summon a demon and bind it: A grounded theory of LLM red teaming. *PLOS ONE* 20 (2023).

[116] 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).

[117] Matthew Jagielski, Giorgio Severi, Niklas Pousette Harger, and Alina Oprea. 2020. Subpopulation Data Poisoning Attacks. *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security* (2020).

[118] 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

[119] Neel Jain, Avi Schwarzschild, Yuxin Wen, Gowthami Somepalli, John Kirchenbauer, Ping yeh Chiang, Micah Goldblum, Aniruddha Saha, Jonas Geiping, and Tom Goldstein. 2023. Baseline Defenses for Adversarial Attacks Against Aligned Language Models. *ArXiv* abs/2309.00614 (2023).

[120] jenkins. 2024. *jenkins.* Retrieved April 20, 2024 from https://www.jenkins.io/

[121] jenkins. 2024. *Jenkins Security Advisory 2024-08-07.* Retrieved April 20, 2024 from https://www.jenkins.io/security/advisory/2024-08-07/

[122] jetbrains. 2024. *PyCharm The Python IDE for data and web professionals.* Retrieved April 20, 2024 from https://www.jetbrains.com/pycharm/

[123] Jiabao Ji, Bairu Hou, Alexander Robey, George J. Pappas, Hamed Hassani, Yang Zhang, Eric Wong, and Shiyu Chang. 2024. Defending Large Language Models against Jailbreak Attacks via Semantic Smoothing. *ArXiv* abs/2402.16192 (2024).

[124] Shuli Jiang, Swanand Ravindra Kadhe, Yi Zhou, Ling Cai, and Nathalie Baracaldo. 2023. Forcing generative models to degenerate ones: The power of data poisoning attacks. *arXiv preprint arXiv:2312.04748* (2023).

[125] Wenxin Jiang, Nicholas 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.

[126] Meenu Mary John, Helena Holmström Olsson, and J. Bosch. 2021. Towards MLOps: A Framework and Maturity Model. *2021 47th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)* (2021), 1–8.

[127] kaggle. 2025. Malicious Prompt Detection Dataset (MPDD). Retrieved January 1, 2026 from https://www.kaggle.com/datasets/mohammedaminejebbar/malicious-prompt-detection-dataset-mpdd

[128] Nikhil Kandpal, Matthew Jagielski, Florian Tramèr, and Nicholas Carlini. 2023. Backdoor Attacks for In-Context Learning with Language Models. *ArXiv* abs/2307.14692 (2023).

[129] Nikhil Kandpal, Krishna Pillutla, Alina Oprea, Peter Kairouz, Christopher A Choquette-Choo, and Zheng Xu. 2023. User inference attacks on large language models. *arXiv preprint arXiv:2310.09266* (2023).

[130] Masahiro Kaneko, Youmi Ma, Yuki Wata, and Naoaki Okazaki. 2024. Sampling-based Pseudo-Likelihood for Membership Inference Attacks. *ArXiv* abs/2404.11262 (2024).

[131] 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.

[132] Aly M. Kassem, Omar Mahmoud, Niloofar Mireshghallah, Hyunwoo Kim, Yulia Tsvetkov, Yejin Choi, Sherif Saad, and Santu Rana. 2024. Alpaca against Vicuna: Using LLMs to Uncover Memorization of LLMs. *ArXiv* abs/2403.04801 (2024).

[133] Siwon Kim, Sangdoo Yun, Hwaran Lee, Martin Gubri, Sung-Hoon Yoon, and Seong Joon Oh. 2023. ProPILE: Probing Privacy Leakage in Large Language Models. *ArXiv* abs/2307.01881 (2023).

[134] 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.

[135] Navya Martin Kollapally and James Geller. 2024. Safeguarding Ethical AI: Detecting Potentially Sensitive Data Re-Identification and Generation of Misleading or Abusive Content from Quantized Large Language Models. In *International Joint Conference on Biomedical Engineering Systems and Technologies*.

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

[137] 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/

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

[139] Aounon Kumar, Chirag Agarwal, Suraj Srinivas, Soheil Feizi, and Himabindu Lakkaraju. 2023. Certifying LLM Safety against Adversarial Prompting. *ArXiv* abs/2309.02705 (2023).

[140] 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).

[141] Divyanshu Kumar, Anurakt Kumar, Sahil Agarwal, and Prashanth Harshangi. 2024. Increased llm vulnerabilities from fine-tuning and quantization. *arXiv preprint arXiv:2404.04392* (2024).

[142] Surender Suresh Kumar, Dr. M.L. Cummings, and Dr. Alexander Stimpson. 2024. Strengthening LLM Trust Boundaries: A Survey of Prompt Injection Attacks Surender Suresh Kumar Dr. M.L. Cummings Dr. Alexander Stimpson. *2024 IEEE 4th International Conference on Human-Machine Systems (ICHMS)* (2024), 1–6.

[143] Vimal Kumar, Juliette Mayo, and Khadija Bahiss. 2024. ADMIn: Attacks on Dataset, Model and Input. A Threat Model for AI Based Software. *ArXiv* abs/2401.07960 (2024).

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

[145] labelyourdata. 2024. *LLM Fine Tuning Tools: For Machine Learning Tasks.* Retrieved April 20, 2024 from https://labelyourdata.com/articles/llm-fine-tuning/top-llm-tools-for-fine-tuning

[146] 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.

[147] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. In *North American Chapter of the Association for Computational Linguistics*.

[148] Katherine Lee, A. Feder Cooper, and James Grimmelmann. 2023. Talkin' 'Bout AI Generation: Copyright and the Generative-AI Supply Chain (The Short Version). *Proceedings of the 2024 Symposium on Computer Science and Law* (2023).

[149] 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).

[150] Boheng Li, Yishuo Cai, Haowei Li, Feng Xue, Zhifeng Li, and Yiming Li. 2024. Nearest is not dearest: Towards practical defense against quantization-conditioned backdoor attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 24523–24533.

[151] Boquan Li, Mengdi Zhang, Peixin Zhang, Jun Sun, and Xingmei Wang. 2024. Resilient Watermarking for LLM-Generated Codes. *ArXiv* abs/2402.07518 (2024).

[152] 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).

[153] Haoran Li, Yulin Chen, Jinglong Luo, Yan Kang, Xiaojin Zhang, Qi Hu, Chunkit Chan, and Yangqiu Song. 2023. Privacy in large language models: Attacks, defenses and future directions. *arXiv preprint arXiv:2310.10383* (2023).

[154] Haoran Li, Dadi Guo, Wei Fan, Mingshi Xu, Jie Huang, and Yangqiu Song. 2023. Multi-step Jailbreaking Privacy Attacks on ChatGPT. *ArXiv* abs/2304.05197 (2023).

[155] Jinmin Li, Kuofeng Gao, Yang Bai, Jingyun Zhang, Shu-Tao Xia, and Yisen Wang. 2024. FMM-Attack: A Flow-based Multi-modal Adversarial Attack on Video-based LLMs. *ArXiv* abs/2403.13507 (2024).

[156] Jie Li, Yi Liu, Chongyang Liu, Ling Shi, Xiaoning Ren, Yaowen Zheng, Yang Liu, and Yinxing Xue. 2024. A Cross-Language Investigation into Jailbreak Attacks in Large Language Models. *ArXiv* abs/2401.16765 (2024).

[157] Jiazhao Li, Yijin Yang, Zhuofeng Wu, V. G. Vinod Vydiswaran, and Chaowei Xiao. 2023. ChatGPT as an Attack Tool: Stealthy Textual Backdoor Attack via Blackbox Generative Model Trigger. In *North American Chapter of the Association for Computational Linguistics*.

[158] 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).

[159] 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.

[160] Wenhao Li, Xiangfeng Wang, Bo Jin, Dijun Luo, and Hongyuan Zha. 2021. Structured cooperative reinforcement learning with time-varying composite action space. *IEEE transactions on pattern analysis and machine intelligence* 44, 11 (2021), 8618–8634.

[161] Xirui Li, Ruochen Wang, Minhao Cheng, Tianyi Zhou, and Cho-Jui Hsieh. 2024. DrAttack: Prompt Decomposition and Reconstruction Makes Powerful LLM Jailbreakers. In *Conference on Empirical Methods in Natural Language Processing*.

[162] Xuan Li, Zhanke Zhou, Jianing Zhu, Jiangchao Yao, Tongliang Liu, and Bo Han. 2023. DeepInception: Hypnotize Large Language Model to Be Jailbreaker. *ArXiv* abs/2311.03191 (2023).

[163] Yuanheng Li, Zhuoyang Chen, Xiaoyun Liu, Yuhao Wang, Mingwei Liu, Yang Shi, Kaifeng Huang, and Shengjie Zhao. 2026. Uncovering Pretraining Code in LLMs: A Syntax-Aware Attribution Approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 40. 721–729.

[164] Yanzhou Li, Tianlin Li, Kangjie Chen, Jian Zhang, Shangqing Liu, Wenhan Wang, Tianwei Zhang, and Yang Liu. 2024. BadEdit: Backdooring large language models by model editing. *ArXiv* abs/2403.13355 (2024).

[165] Yansong Li, Zhixing Tan, and Yang Liu. 2023. Privacy-Preserving Prompt Tuning for Large Language Model Services. *ArXiv* abs/2305.06212 (2023).

[166] Yuanchun Li, Hao Wen, Weijun Wang, Xiangyu Li, Yizhen Yuan, Guohong Liu, Jiacheng Liu, Wenxing Xu, Xiang Wang, Yi Sun, Rui Kong, Yile Wang, Hanfei Geng, Jian Luan, Xuefeng Jin, Zi-Liang Ye, Guanjing Xiong, Fan Zhang, Xiang Li, Mengwei Xu, Zhijun Li, Peng Li, Yang Liu, Yaqiong Zhang, and Yunxin Liu. 2024. Personal LLM Agents: Insights and Survey about the Capability, Efficiency and Security. *ArXiv* abs/2401.05459 (2024).

[167] Jiawei Liang, Siyuan Liang, Aishan Liu, and Xiaochun Cao. 2025. VL-Trojan: Multimodal Instruction Backdoor Attacks against Autoregressive Visual Language Models. *Int. J. Comput. Vis.* 133 (2025), 3994–4013.

[168] Jiayi Liang, Xi Zhang, Yuming Shang, Sanchuan Guo, and Chaozhuo Li. 2023. Clean-label Poisoning Attack against Fake News Detection Models. *2023 IEEE International Conference on Big Data (BigData)* (2023), 3614–3623.

[169] 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.

[170] linkedin. 2025. Hugging Face Secrets Leak Highlights AI Supply Chain Risk. Retrieved January 1, 2026 from https://www.linkedin.com/pulse/hugging-face-secrets-leak-highlights-ai-supply-chain-risk-h5a2e/

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

[172] Shuhan Liu, Xing Hu, Xin Xia, David Lo, and Xiaohu Yang. 2025. An Empirical Study of Vulnerable Package Dependencies in LLM Repositories. *arXiv preprint arXiv:2508.21417* (2025).

[173] Xiaoyu Liu, Paiheng Xu, Junda Wu, Jiaxin Yuan, Yifan Yang, Yuhang Zhou, Fuxiao Liu, Tianrui Guan, Haoliang Wang, Tong Yu, Julian J. McAuley, Wei Ai, and Furong Huang. 2025. Large Language Models and Causal Inference in Collaboration: A Comprehensive Survey. *ArXiv* abs/2403.09606 (2025).

[174] Xiaogeng Liu, Zhiyuan Yu, Yizhe Zhang, Ning Zhang, and Chaowei Xiao. 2024. Automatic and Universal Prompt Injection Attacks against Large Language Models. *ArXiv* abs/2403.04957 (2024).

[175] Xinyu Liu, Yukai Zhao, Xing Hu, and Xin Xia. 2026. Exploiting LLM Agent Supply Chains via Payload-less Skills. *arXiv preprint arXiv:2605.14460* (2026).

[176] Yi Liu, Gelei Deng, Yuekang Li, Kailong Wang, Tianwei Zhang, Yepang Liu, Haoyu Wang, Yanhong Zheng, and Yang Liu. 2023. Prompt Injection attack against LLM-integrated Applications. *ArXiv* abs/2306.05499 (2023).

[177] Yi Liu, Gelei Deng, Zhengzi Xu, Yuekang Li, Yaowen Zheng, Ying Zhang, Lida Zhao, Tianwei Zhang, and Yang Liu. 2023. Jailbreaking ChatGPT via Prompt Engineering: An Empirical Study. *ArXiv* abs/2305.13860 (2023).

[178] Yue Liu, Dawen Zhang, Boming Xia, Julia Anticev, Tunde Adebayo, Zhenchang Xing, and Moses Machao. 2024. Blockchain-Enabled Accountability in Data Supply Chain: A Data Bill of Materials Approach. *2024 IEEE International Conference on Blockchain (Blockchain)* (2024), 557–562.

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

[180] llmmodels. 2024. *Version Control for Large Language Models: Step-by-Step Guide.* Retrieved April 20, 2024 from https://llmmodels.org/blog/version-control-for-large-language-models-step-by-step-guide/

[181] Nicola Lucchi. 2023. ChatGPT: A Case Study on Copyright Challenges for Generative Artificial Intelligence Systems. *European Journal of Risk Regulation* 15 (2023), 602 – 624.

[182] Weidi Luo, Siyuan Ma, Xiaogeng Liu, Xiaoyu Guo, and Chaowei Xiao. 2024. JailBreakV: A Benchmark for Assessing the Robustness of MultiModal Large Language Models against Jailbreak Attacks.

[183] 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).

[184] Yuzhe Ma, Xiaojin Zhu, and Justin Hsu. 2019. Data Poisoning against Differentially-Private Learners: Attacks and Defenses. In *International Joint Conference on Artificial Intelligence*.

[185] Pratyush Maini, Hengrui Jia, Nicolas Papernot, and Adam Dziedzic. 2024. LLM Dataset Inference: Did you train on my dataset? *Advances in Neural Information Processing Systems* 37 (2024), 124069–124092.

[186] Neal Mangaokar, Ashish Hooda, Jihye Choi, Shreyas Chandrashekaran, Kassem Fawaz, Somesh Jha, and Atul Prakash. 2024. PRP: Propagating Universal Perturbations to Attack Large Language Model Guard-Rails. *ArXiv* abs/2402.15911 (2024).

[187] Marc Marone and Benjamin Van Durme. 2023. Data Portraits: Recording Foundation Model Training Data. *ArXiv* abs/2303.03919 (2023).

[188] 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*.

[189] Timothy R. Mcintosh, Teo Susnjak, Tong Liu, Paul A. Watters, and Malka N. Halgamuge. 2024. The Inadequacy of Reinforcement Learning From Human Feedback—Radicalizing Large Language Models via Semantic Vulnerabilities. *IEEE Transactions on Cognitive and Developmental Systems* 16 (2024), 1561–1574.

[190] medicaleconomics. 2025. Health care workers are leaking patient data through AI tools, cloud apps. Retrieved January 1, 2026 from https://www.medicaleconomics.com/view/health-care-workers-are-leaking-patient-data-through-ai-tools-cloud-apps

[191] medium. 2024. *The Rise of Model Serving Frameworks: Why Triton Inference Server Matters.* Retrieved April 20, 2024 from https://medium.com/python-and-machine-learning-pearls/the-rise-of-model-serving-frameworks-why-triton-inference-server-matters-8ef3eab372e0

[192] Matthieu Meeus, Igor Shilov, Manuel Faysse, and Yves-Alexandre de Montjoye. 2024. Copyright Traps for Large Language Models. *ArXiv* abs/2402.09363 (2024).

[193] Anay Mehrotra, Manolis Zampetakis, Paul Kassianik, Blaine Nelson, Hyrum Anderson, Yaron Singer, and Amin Karbasi. 2023. Tree of Attacks: Jailbreaking Black-Box LLMs Automatically. *ArXiv* abs/2312.02119 (2023).

[194] Kai Mei, Zheng Li, Zhenting Wang, Yang Zhang, and Shiqing Ma. 2023. NOTABLE: Transferable Backdoor Attacks Against Prompt-based NLP Models. In *Annual Meeting of the Association for Computational Linguistics*.

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

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

[197] Dmitry Mikhailov. 2023. Optimizing National Security Strategies through LLM-Driven Artificial Intelligence Integration. *ArXiv* abs/2305.13927 (2023).

[198] Yichuan Mo, Yuji Wang, Zeming Wei, and Yisen Wang. 2024. Fight Back Against Jailbreaking via Prompt Adversarial Tuning. In *Neural Information Processing Systems*.

[199] 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

[200] Silen Naihin, David Atkinson, Marc Green, Merwane Hamadi, Craig Swift, Douglas Schonholtz, Adam Tauman Kalai, and David Bau. 2023. Testing Language Model Agents Safely in the Wild. *ArXiv* abs/2311.10538 (2023).

[201] Takeru Naito, Rei Watanabe, and Takuho Mitsunaga. 2023. LLM-based Attack Scenarios Generator with IT Asset Management and Vulnerability Information. *2023 6th International Conference on Signal Processing and Information Security (ICSPIS)* (2023), 99–103.

[202] Assaf Namer, Prashant Kulkarni, Erik Jeansson, Brandon Maltzman, and Hauke Vagts. 2024. Automatically Detecting Expensive Prompts and Configuring Firewall Rules to Mitigate Denial of Service Attacks on Large Language Models. (2024).

[203] Assaf Namer, Hauke Vagts, Jim Miller, Brandon Maltzman, and Guy Rinkevich. 2024. Training Dataset Validation to Protect Machine Learning Models from Data Poisoning. (2024).

[204] Najmeh Nazari, Furi Xiang, Chongzhou Fang, Hosein Mohammadi Makrani, Aditya Puri, Kartik Patwari, Hossein Sayadi, Setareh Rafatirad, Chen-Nee Chuah, and Houman Homayoun. 2024. LLM-FIN: Large Language Models Fingerprinting Attack on Edge Devices. *2024 25th International Symposium on Quality Electronic Design (ISQED)* (2024), 1–6.

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

[206] neptune. 2024. *Hyperparameter Tuning in Python: a Complete Guide.* Retrieved April 20, 2024 from https://neptune.ai/blog/hyperparameter-tuning-in-python-complete-guide

[207] 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

[208] 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

[209] 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

[210] Zhenxing Niu, Haodong Ren, Xinbo Gao, Gang Hua, and Rong Jin. 2024. Jailbreaking Attack against Multimodal Large Language Model. *ArXiv* abs/2402.02309 (2024).

[211] NVIDIA. 2025. *NVIDIA H100 GPU.* Retrieved December 20, 2025 from https://www.nvidia.com/en-us/data-center/h100/

[212] NVIDIA. 2025. *NVIDIA NVLink and NVLink Switch.* Retrieved December 20, 2025 from https://www.nvidia.com/en-us/data-center/nvlink/

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

[214] oligo. 2024. *More Models, More ProbLLMs.* Retrieved April 20, 2024 from https://www.oligo.security/blog/more-models-more-probllms

[215] oligo. 2024. *ShadowRay: First Known Attack Campaign Targeting AI Workloads Actively Exploited In The Wild.* Retrieved April 20, 2024 from https://www.oligo.security/blog/shadowray-attack-ai-workloads-actively-exploited-in-the-wild

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

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

[218] openai. 2025. Experiencing Decreased Performance with ChatGPT-4. Retrieved January 1, 2026 from https://community.openai.com/t/experiencing-decreased-performance-with-chatgpt-4/234269/3

[219] 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.

[220] Tinghui Ouyang, Hoang-Quoc Nguyen-Son, Huy Hoang Nguyen, Isao Echizen, and Yoshiki Seo. 2023. Quality Assurance of A GPT-Based Sentiment Analysis System: Adversarial Review Data Generation and Detection. *2023 30th Asia-Pacific Software Engineering Conference (APSEC)* (2023), 450–457.

[221] 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).

[222] Yin Minn Pa Pa, Shunsuke Tanizaki, Tetsui Kou, Michel van Eeten, Katsunari Yoshioka, and Tsutomu Matsumoto. 2023. An Attacker's Dream? Exploring the Capabilities of ChatGPT for Developing Malware. *Proceedings of the 16th Cyber Security Experimentation and Test Workshop* (2023).

[223] Anwesan Pal, Radhika Bhargava, Kyle Hinsz, Jacques Esterhuizen, and Sudipta Bhattacharya. 2024. The Empirical Impact of Data Sanitization on Language Models. *ArXiv* abs/2411.05978 (2024).

[224] 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.

[225] 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.

[226] Yikang Pan, Liangming Pan, Wenhu Chen, Preslav Nakov, Min-Yen Kan, and William Yang Wang. 2023. On the risk of misinformation pollution with large language models. *arXiv preprint arXiv:2305.13661* (2023).

[227] Ashwinee Panda, Christopher A. Choquette-Choo, Zhengming Zhang, Yaoqing Yang, and Prateek Mittal. 2024. Teach LLMs to Phish: Stealing Private Information from Language Models. *ArXiv* abs/2403.00871 (2024).

[228] Qi Pang, Shengyuan Hu, Wenting Zheng, and Virginia Smith. 2024. Attacking llm watermarks by exploiting their strengths. In *ICLR 2024 Workshop on Secure and Trustworthy Large Language Models*.

[229] 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).

[230] Vaidehi Patil, Peter Hase, and Mohit Bansal. 2023. Can Sensitive Information Be Deleted From LLMs? Objectives for Defending Against Extraction Attacks. *ArXiv* abs/2309.17410 (2023).

[231] Martin Pawelczyk, Seth Neel, and Himabindu Lakkaraju. 2023. In-Context Unlearning: Language Models as Few Shot Unlearners. *ArXiv* abs/2310.07579 (2023).

[232] Julien Piet, Maha Alrashed, Chawin Sitawarin, Sizhe Chen, Zeming Wei, Elizabeth Sun, Basel Alomair, and David Wagner. 2023. Jatmo: Prompt Injection Defense by Task-Specific Finetuning. In *European Symposium on Research in Computer Security*.

[233] Matthew Pisano, Peter Ly, Abraham Sanders, Bingsheng Yao, Dakuo Wang, Tomek Strzalkowski, and Mei Si. 2023. Bergeron: Combating Adversarial Attacks through a Conscience-Based Alignment Framework. *ArXiv* abs/2312.00029 (2023).

[234] 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/

[235] 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/

[236] 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.

[237] Gopi Krishnan Rajbahadur, Keheliya Gallaba, Elyas Rashno, Arthit Suriyawongkul, Karen Bennet, Kate Stewart, and Ahmed E. Hassan. 2025. Building an Open AIBOM Standard in the Wild. *ArXiv* abs/2510.07070 (2025).

[238] Javier Rando and Florian Tramèr. 2023. Universal Jailbreak Backdoors from Poisoned Human Feedback. *ArXiv* abs/2311.14455 (2023).

[239] Abhinav Rao, Sachin Vashistha, Atharva Naik, Somak Aditya, and Monojit Choudhury. 2023. Tricking LLMs into Disobedience: Understanding, Analyzing, and Preventing Jailbreaks. *ArXiv* abs/2305.14965 (2023).

[240] Md. Rafi Ur Rashid, Vishnu Asutosh Dasu, Kang Gu, Najrin Sultana, and Shagufta Mehnaz. 2023. FLTrojan: Privacy Leakage Attacks against Federated Language Models Through Selective Weight Tampering. *ArXiv* abs/2310.16152 (2023).

[241] reuters. 2025. *OpenAI, Microsoft want court to toss lawsuit accusing them of abusing open-source code.* Retrieved December 20, 2025 from https://www.reuters.com/legal/litigation/openai-microsoft-want-court-toss-lawsuit-accusing-them-abusing-open-source-code-2023-01-27/

[242] reversinglabs. 2024. *Malicious ML models discovered on Hugging Face platform.* Retrieved April 20, 2024 from https://www.reversinglabs.com/blog/rl-identifies-malware-ml-model-hosted-on-hugging-face

[243] 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).

[244] Jiaqi Ruan, Gaoqi Liang, Huan Zhao, Guolong Liu, Xianzhuo Sun, Jing Qiu, Zhao Xu, Fushuan Wen, and Zhao Yang Dong. 2023. Applying Large Language Models to Power Systems: Potential Security Threats. *IEEE Transactions on Smart Grid* 15 (2023), 3333–3336.

[245] Mark Russinovich, Ahmed Salem, and Ronen Eldan. 2024. Great, Now Write an Article About That: The Crescendo Multi-Turn LLM Jailbreak Attack. *ArXiv* abs/2404.01833 (2024).

[246] Ahmed Salem, Andrew J. Paverd, and Boris Köpf. 2023. Maatphor: Automated Variant Analysis for Prompt Injection Attacks. *ArXiv* abs/2312.11513 (2023).

[247] Iqbal H Sarker. 2024. LLM potentiality and awareness: a position paper from the perspective of trustworthy and responsible AI modeling. *Discover Artificial Intelligence* 4, 1 (2024), 40.

[248] scademy. 2025. Model Extraction Attacks: An Emerging Threat to AI Systems. Retrieved January 1, 2026 from https://www.scademy.ai/post/model-extraction-attacks-an-emerging-threat-to-ai-systems

[249] Sander Schulhoff, Jeremy Pinto, Anaum Khan, Louis-Franccois Bouchard, Chenglei Si, Svetlina Anati, Valen Tagliabue, Anson Liu Kost, Christopher Carnahan, and Jordan L. Boyd-Graber. 2023. Ignore This Title and HackAPrompt: Exposing Systemic Vulnerabilities of LLMs Through a Global Prompt Hacking Competition. In *Conference on Empirical Methods in Natural Language Processing*.

[250] Avi Schwarzschild, Micah Goldblum, Arjun Gupta, John P. Dickerson, and Tom Goldstein. 2020. Just How Toxic is Data Poisoning? A Unified Benchmark for Backdoor and Data Poisoning Attacks. *ArXiv* abs/2006.12557 (2020).

[251] Leo Schwinn, David Dobre, Stephan Günnemann, and Gauthier Gidel. 2023. Adversarial Attacks and Defenses in Large Language Models: Old and New Threats. *ArXiv* abs/2310.19737 (2023).

[252] Leo Schwinn, David Dobre, Sophie Xhonneux, Gauthier Gidel, and Stephan Günnemann. 2024. Soft Prompt Threats: Attacking Safety Alignment and Unlearning in Open-Source LLMs through the Embedding Space. *ArXiv* abs/2402.09063 (2024).

[253] 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

[254] scmagazine. 2024. *SAP AI Core flaws show risks of training Alin 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

[255] security. 2025. Security Flaws within ChatGPT Ecosystem Allowed Access to Accounts On Third-Party Websites and Sensitive Data. Retrieved January 1, 2026 from https://salt.security/blog/security-flaws-within-chatgpt-extensions-allowed-access-to-accounts-on-third-party-websites-and-sensitive-data

[256] securityweek. 2024. *Fake DeepSeek Sites Used for Credential Phishing, Crypto Theft, Scams.* Retrieved April 20, 2024 from https://www.securityweek.com/fake-deepseek-sites-used-for-credential-phishing-crypto-theft-scams/

[257] securityweek. 2024. *Ray AI Framework Vulnerability Exploited to Hack Hundreds of Clusters.* Retrieved April 20, 2024 from https://www.securityweek.com/attackers-exploit-ray-ai-framework-vulnerability-to-hack-hundreds-of-clusters/

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

[259] Konstantinos Sgantzos, Mohamed Al Hemairy, Panagiotis Tzavaras, and Spyridon Stelios. 2023. Triple-Entry Accounting as a Means of Auditing Large Language Models. *Journal of Risk and Financial Management* (2023).

[260] Zeyang Sha and Yang Zhang. 2024. Prompt Stealing Attacks Against Large Language Models. *ArXiv* abs/2402.12959 (2024).

[261] Shlomit Shachor, Natalia Razinkov, and Abigail Goldsteen. 2023. Improved Membership Inference Attacks Against Language Classification Models. *ArXiv* abs/2310.07219 (2023).

[262] Rusheb Shah, Quentin Feuillade-Montixi, Soroush Pour, Arush Tagade, Stephen Casper, and Javier Rando. 2023. Scalable and Transferable Black-Box Jailbreaks for Language Models via Persona Modulation. *ArXiv* abs/2311.03348 (2023).

[263] Rusheb Shah, Quentin Feuillade Montixi, Soroush Pour, Arush Tagade, and Javier Rando. [n. d.]. Jailbreaking Language Models at Scale via Persona Modulation. ([n. d.]).

[264] 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.

[265] Reshabh K Sharma, Vinayak Gupta, and Dan Grossman. 2024. SPML: A DSL for Defending Language Models Against Prompt Attacks. *ArXiv* abs/2402.11755 (2024).

[266] 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*.

[267] Erfan Shayegani, Md Abdullah Al Mamun, Yu Fu, Pedram Zaree, Yue Dong, and Nael Abu-Ghazaleh. 2023. Survey of vulnerabilities in large language models revealed by adversarial attacks. *arXiv preprint arXiv:2310.10844* (2023).

[268] 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. *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security* (2023).

[269] Jiawen Shi, Zenghui Yuan, Yinuo Liu, Yue Huang, Pan Zhou, Lichao Sun, and Neil Zhenqiang Gong. 2024. Optimization-based Prompt Injection Attack to LLM-as-a-Judge. *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security* (2024).

[270] Dong Shu, Mingyu Jin, Suiyuan Zhu, Beichen Wang, Zihao Zhou, Chong Zhang, and Yongfeng Zhang. 2024. AttackEval: How to Evaluate the Effectiveness of Jailbreak Attacking on Large Language Models. *ArXiv* abs/2401.09002 (2024).

[271] 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).

[272] 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.

[273] 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).

[274] Mohammed Latif Siddiq, Jiahao Zhang, Lindsay Roney, and Joanna C. S. Santos. 2024. Re(gEx|DoS)Eval: Evaluating Generated Regular Expressions and their Proneness to DoS Attacks. *2024 IEEE/ACM 46th International Conference on Software Engineering: New Ideas and Emerging Results (ICSE-NIER)* (2024), 52–56.

[275] Mohammed Latif Siddiq, Jiahao Zhang, and Joanna C. S. Santos. 2024. Understanding Regular Expression Denial of Service (ReDoS): Insights from LLM-Generated Regexes and Developer Forums. *2024 IEEE/ACM 32nd International Conference on Program Comprehension (ICPC)* (2024), 190–201.

[276] Shridhar Singh. 2024. Enhancing Privacy and Security in Large-Language Models: A Zero-Knowledge Proof Approach. *International Conference on Cyber Warfare and Security* (2024).

[277] Tanmay Singh, Harshvardhan Aditya, Vijay Krishna Madisetti, and Arshdeep Bahga. 2024. Whispered Tuning: Data Privacy Preservation in Fine-Tuning LLMs through Differential Privacy. *Journal of Software Engineering and Applications* (2024).

[278] Chawin Sitawarin, Norman Mu, David Wagner, and Alexandre Araujo. 2024. PAL: Proxy-Guided Black-Box Attack on Large Language Models. *ArXiv* abs/2402.09674 (2024).

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

[280] Tyler Sorensen and Heidy Khlaaf. 2024. LeftoverLocals: Listening to LLM Responses Through Leaked GPU Local Memory. *ArXiv* abs/2401.16603 (2024).

[281] Alexandra Souly, Javier Rando, Ed Chapman, Xander Davies, Burak Hasircioglu, Ezzeldin Shereen, Carlos Mougan, Vasilios Mavroudis, Erik Jones, Chris Hicks, et al. 2025. Poisoning attacks on LLMs require a near-constant number of poison samples. *arXiv preprint arXiv:2510.07192* (2025).

[282] Tobin South, Robert Mahari, and Thomas Hardjono. [n. d.]. Secure Community Transformers: Private Pooled Data for LLMs.

[283] spdx. 2025. *SPDX AIBOM*. Retrieved December 20, 2025 from https://spdx.dev/implementing-an-ai-bom/

[284] spdx. 2025. *SPDX SBOM*. Retrieved December 20, 2025 from https://spdx.github.io/spdx-spec/v3.0.1/model/Software/Classes/Sbom/

[285] Joseph Squillace, Justice Cappella, and Andrew Sepp. 2024. User Vulnerabilities in AI-Driven Systems: Current Cybersecurity Threat Dynamics and Malicious Exploits in Supply Chain Management and Project Management. *2024 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems (ICETSIS)* (2024), 1760–1765.

[286] 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*.

[287] Trevor Stalnaker. 2025. Understanding and Supporting the ML Supply Chain Through ML Bill of Materials. *2025 IEEE/ACM 47th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)* (2025), 1–3.

[288] Trevor Stalnaker, Nathan Wintersgill, Oscar Chaparro, Laura A. Heymann, Massimiliano Di Penta, Daniel M German, and Denys Poshyvanyk. 2024. Developer Perspectives on Licensing and Copyright Issues Arising from Generative AI for Software Development. *ACM Transactions on Software Engineering and Methodology* (2024).

[289] Jacob Steinhardt, Pang Wei Koh, and Percy Liang. 2017. Certified Defenses for Data Poisoning Attacks. In *Neural Information Processing Systems*.

[290] Mahesh Subedar, Nilesh A. Ahuja, Ranganath Krishnan, Ibrahima J. Ndiour, and Omesh Tickoo. 2019. Deep Probabilistic Models to Detect Data Poisoning Attacks. *ArXiv* abs/1912.01206 (2019).

[291] 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.

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

[293] Zhen Tan, Chengshuai Zhao, Raha Moraffah, Yifan Li, Yu Kong, Tianlong Chen, and Huan Liu. 2024. The Wolf Within: Covert Injection of Malice into MLLM Societies via an MLLM Operative. *ArXiv* abs/2402.14859 (2024).

[294] 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.

[295] Xijia Tao, Shuai Zhong, Lei Li, Qi Liu, and Lingpeng Kong. 2024. ImgTrojan: Jailbreaking Vision-Language Models with ONE Image. *ArXiv* abs/2403.02910 (2024).

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

[297] techradar. 2024. *Meta Llama LLM security flaw could let hackers easily breach systems and spread malware*. Retrieved April 20, 2024 from https://www.techradar.com/pro/security/meta-llama-llm-security-flaw-could-let-hackers-easily-breach-systems-and-spread-malware

[298] thehackernews. 2024. *Experts Uncover Severe AWS Flaws Leading to RCE, Data Theft, and Full-Service Takeovers*. Retrieved April 20, 2024 from https://thehackernews.com/2024/08/experts-uncover-severe-aws-flaws.html

[299] thehackernews. 2025. AWS Patches Critical 'FlowFixation' Bug in Airflow Service to Prevent Session Hijacking. Retrieved January 1, 2026 from https://thehackernews.com/2024/03/aws-patches-critical-flowfixation-bug.html

[300] theregister. 2024. *Crooks stole AWS credentials from misconfigured sites then kept them in open S3 bucket.* Retrieved April 20, 2024 from https://www.theregister.com/2024/12/09/aws_credentials_stolen/

[301] Saliltorn Thongmeensuk. 2024. Rethinking copyright exceptions in the era of generative AI: Balancing innovation and intellectual property protection. *The Journal of World Intellectual Property* (2024).

[302] Yu Tian, Xiao Yang, Yinpeng Dong, Heming Yang, Hang Su, and Jun Zhu. 2024. BSPA: Exploring Black-box Stealthy Prompt Attacks against Image Generators. *ArXiv* abs/2402.15218 (2024).

[303] Yu Tian, Xiao Yang, Jingyuan Zhang, Yinpeng Dong, and Hang Su. 2023. Evil Geniuses: Delving into the Safety of LLM-based Agents. *ArXiv* abs/2311.11855 (2023).

[304] 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.

[305] Catherine Tony, Markus Mutas, Nicolás E. Díaz Ferreyra, and Riccardo Scandariato. 2023. LLMSecEval: A Dataset of Natural Language Prompts for Security Evaluations. *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)* (2023), 588–592.

[306] Dale Townsend. 2021. Validation and inference of agent based models. *arXiv preprint arXiv:2107.03619* (2021).

[307] trustai. 2024. *llm apps plugin in ddos risk dont.* Retrieved April 20, 2024 from https://trustai.sg/2024/09/06/llm-apps-plugin-in-ddos-risk-dont/

[308] Andreas Tsamados, L. Floridi, and Mariarosaria Taddeo. 2023. The Cybersecurity Crisis of Artificial Intelligence: Unrestrained Adoption and Natural Language-Based Attacks. *ArXiv* abs/2311.09224 (2023).

[309] Imdad Ullah, Najm Hassan, Sukhpal Singh Gill, Basem Suleiman, Tariq Ahamed Ahanger, Zawar Shah, Junaid Qadir, and Salil S. Kanhere. 2023. Privacy Preserving Large Language Models: ChatGPT Case Study Based Vision and Framework. *IET Blockchain* 4 (2023), 706–724.

[310] Jason Vega, Isha Chaudhary, Changming Xu, and Gagandeep Singh. 2023. Bypassing the Safety Training of Open-Source LLMs with Priming Attacks. *ArXiv* abs/2312.12321 (2023).

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

[312] visualstudio. 2024. *Visual Studio Code.* Retrieved April 20, 2024 from https://code.visualstudio.com/

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

[314] 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*.

[315] Chaofan Wang, Samuel Kernan Freire, Mo Zhang, Jing Wei, Jorge Gonçalves, Vassilis Kostakos, Zhanna Sarsenbayeva, Christina Schneegass, Alessandro Bozzon, and Evangelos Niforatos. 2023. Safeguarding Crowdsourcing Surveys from ChatGPT with Prompt Injection. *ArXiv* abs/2306.08833 (2023).

[316] Guanyu Wang, Yuekang Li, Yi Liu, Gelei Deng, Tianlin Li, Guosheng Xu, Yang Liu, Haoyu Wang, and Kailong Wang. 2024. MeTMaP: Metamorphic Testing for Detecting False Vector Matching Problems in LLM Augmented Generation. *2024 IEEE/ACM First International Conference on AI Foundation Models and Software Engineering (Forge) Conference Acronym:* (2024), 12–23.

[317] Guangjing Wang, Ce Zhou, Yuanda Wang, Bocheng Chen, Hanqing Guo, and Qiben Yan. 2023. Beyond Boundaries: A Comprehensive Survey of Transferable Attacks on AI Systems. *ArXiv* abs/2311.11796 (2023).

[318] Hao Wang, Hao Li, Minlie Huang, and Lei Sha. 2024. From Noise to Clarity: Unraveling the Adversarial Suffix of Large Language Model Attacks via Translation of Text Embeddings. *ArXiv* abs/2402.16006 (2024).

[319] Haoran Wang and Kai Shu. 2023. Backdoor Activation Attack: Attack Large Language Models using Activation Steering for Safety-Alignment. *ArXiv* abs/2311.09433 (2023).

[320] Jingtan Wang, Xinyang Lu, Zitong Zhao, Zhongxiang Dai, Chuan-Sheng Foo, See-Kiong Ng, and Kian Hsiang Low. 2025. WASA: WAtermark-based Source Attribution for Large Language Model-Generated Data. In *Annual Meeting of the Association for Computational Linguistics*.

[321] Jiongxiao Wang, Junlin Wu, Muhao Chen, Yevgeniy Vorobeychik, and Chaowei Xiao. 2024. Rlhfpoison: Reward poisoning attack for reinforcement learning with human feedback in large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2551–2570.

[322] 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).

[323] Meng Wang, Ningyu Zhang, Ziwen Xu, Zekun Xi, Shumin Deng, Yunzhi Yao, Qishen Zhang, Linyi Yang, Jindong Wang, and Huajun Chen. 2024. Detoxifying Large Language Models via Knowledge Editing. *ArXiv* abs/2403.14472 (2024).

[324] Pengyu Wang, Dong Zhang, Linyang Li, Chenkun Tan, Xinghao Wang, Ke Ren, Botian Jiang, and Xipeng Qiu. 2024. InferAligner: Inference-Time Alignment for Harmlessness through Cross-Model Guidance. In *Conference on Empirical Methods in Natural Language Processing*.

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

[326] Xunguang Wang, Zhenlan Ji, Pingchuan Ma, Zongjie Li, and Shuai Wang. 2023. InstructTA: Instruction-Tuned Targeted Attack for Large Vision-Language Models. *ArXiv* abs/2312.01886 (2023).

[327] Yu Wang, Xiaogeng Liu, Yu Li, Muhao Chen, and Chaowei Xiao. 2024. AdaShield: Safeguarding Multimodal Large Language Models from Structure-based Attack via Adaptive Shield Prompting. In *European Conference on Computer Vision*.

[328] Zhilong Wang, Yebo Cao, and Peng Liu. 2024. Hidden You Malicious Goal Into Benign Narratives: Jailbreak Large Language Models through Logic Chain Injection. *ArXiv* abs/2404.04849 (2024).

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

[330] Zezhong Wang, Fangkai Yang, Lu Wang, Pu Zhao, Hongru Wang, Liang Chen, Qingwei Lin, and Kam-Fai Wong. 2023. SELF-GUARD: Empower the LLM to Safeguard Itself. *ArXiv* abs/2310.15851 (2023).

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

[332] Roy Weiss, Daniel Ayzenshteyn, Guy Amit, and Yisroel Mirsky. 2024. What Was Your Prompt? A Remote Keylogging Attack on AI Assistants. *ArXiv* abs/2403.09751 (2024).

[333] wikipedia. 2024. *XcodeGhost*. Retrieved April 20, 2024 from https://en.wikipedia.org/wiki/XcodeGhost

[334] 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/

[335] Claes Wohlin. 2014. Guidelines for snowballing in systematic literature studies and a replication in software engineering. In *Proceedings of the 18th international conference on evaluation and assessment in software engineering*. 1–10.

[336] Fangzhou Wu, Xiaogeng Liu, and Chaowei Xiao. 2023. DeceptPrompt: Exploiting LLM-driven Code Generation via Adversarial Natural Language Instructions. *ArXiv* abs/2312.04730 (2023).

[337] Fangzhou Wu, Shutong Wu, Yulong Cao, and Chaowei Xiao. 2024. WIPI: A New Web Threat for LLM-Driven Web Agents. *ArXiv* abs/2402.16965 (2024).

[338] 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).

[339] Susheng Wu, Ziqian Chen, Chengyuan Li, Kaifeng Huang, Zekai Chen, Yijian Wu, Bihuan Chen, Yiheng Cao, Zhuotong Zhou, Yiheng Huang, and Xin Peng. 2026. TensorLock: Recovering Model Dependency for Model Supply Chain. In *Proceedings of the 35th ACM SIGSOFT International Symposium on Software Testing and Analysis*.

[340] Xiyang Wu, Ruiqi Xian, Tianrui Guan, Jing Liang, Souradip Chakraborty, Fuxiao Liu, Brian M. Sadler, Dinesh Manocha, and A. S. Bedi. 2024. On the Safety Concerns of Deploying LLMs/VLMs in Robotics: Highlighting the Risks and Vulnerabilities. *ArXiv* abs/2402.10340 (2024).

[341] Yuhao Wu, Franziska Roesner, Tadayoshi Kohno, Ning Zhang, and Umar Iqbal. 2024. IsolateGPT: An Execution Isolation Architecture for LLM-Based Agentic Systems. In *Network and Distributed System Security Symposium*.

[342] 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).

[343] Yueqi Xie, Jingwei Yi, Jiawei Shao, Justin Curl, Lingjuan Lyu, Qifeng Chen, Xing Xie, and Fangzhao Wu. 2023. Defending ChatGPT against jailbreak attack via self-reminders. *Nature Machine Intelligence* 5 (2023), 1486–1496.

[344] Jun Xu. 2022. MLOps in the financial industry: Philosophy practices and tools. *Future and Fintech, the, Abcdi and Beyond* (2022), 451.

[345] Lei Xu, Yangyi Chen, Ganqu Cui, Hongcheng Gao, and Zhiyuan Liu. 2022. Exploring the Universal Vulnerability of Prompt-based Learning Paradigm. In *NAACL-HLT*.

[346] Qiongkai Xu, Jun Wang, Olga Ohrimenko, and Trevor Cohn. 2023. FLAT-Chat: A Word Recovery Attack on Federated Language Model Training. (2023).

[347] Xilie Xu, Keyi Kong, Ninghao Liu, Li zhen Cui, Di Wang, Jingfeng Zhang, and Mohan S. Kankanhalli. 2023. An LLM can Fool Itself: A Prompt-Based Adversarial Attack. *ArXiv* abs/2310.13345 (2023).

[348] Xiaojun Xu, Yuanshun Yao, and Yang Liu. 2024. Learning to Watermark LLM-generated Text via Reinforcement Learning. *ArXiv* abs/2403.10553 (2024).

[349] Yuancheng Xu, Jiarui Yao, Manli Shu, Yanchao Sun, Zichu Wu, Ning Yu, Tom Goldstein, and Furong Huang. 2024. Shadowcast: Stealthy Data Poisoning Attacks Against Vision-Language Models. *ArXiv* abs/2402.06659 (2024).

[350] Zhangchen Xu, Fengqing Jiang, Luyao Niu, Jinyuan Jia, Bill Yuchen Lin, and Radha Poovendran. 2024. SafeDecoding: Defending against Jailbreak Attacks via Safety-Aware Decoding. *ArXiv* abs/2402.08983 (2024).

[351] Zihao Xu, Yi Liu, Gelei Deng, Yuekang Li, and Stjepan Picek. 2024. LLM Jailbreak Attack versus Defense Techniques - A Comprehensive Study. *ArXiv* abs/2402.13457 (2024).

[352] Jiaqi Xue, Yepeng Liu, Mengxin Zheng, Ting Hua, Yilin Shen, Ladislau Boloni, and Qian Lou. 2023. TrojLLM: A Black-box Trojan Prompt Attack on Large Language Models. In *Neural Information Processing Systems*.

[353] Biwei Yan, Kun Li, Minghui Xu, Yueyan Dong, Yue Zhang, Zhaochun Ren, and Xiuzhen Cheng. 2024. On Protecting the Data Privacy of Large Language Models (LLMs): A Survey. *2024 International Conference on Meta Computing (ICMC)* (2024), 1–12.

[354] Jun Yan, Vikas Yadav, SHIYANG LI, Lichang Chen, Zheng Tang, Hai Wang, Vijay Srinivasan, Xiang Ren, and Hongxia Jin. 2023. Backdooring Instruction-Tuned Large Language Models with Virtual Prompt Injection. In *North American Chapter of the Association for Computational Linguistics*.

[355] Yikuan Yan, Yaolun Zhang, and Keman Huang. 2024. Depending on yourself when you should: Mentoring LLM with RL agents to become the master in cybersecurity games. *ArXiv* abs/2403.17674 (2024).

[356] Haomiao Yang, Kunlan Xiang, Mengyu Ge, Hongwei Li, Rongxing Lu, and Shui Yu. 2023. A Comprehensive Overview of Backdoor Attacks in Large Language Models Within Communication Networks. *IEEE Network* 38 (2023), 211–218.

[357] Shu Yang, Jiayuan Su, Han Jiang, Mengdi Li, Keyuan Cheng, Muhammad Asif Ali, Lijie Hu, and Di Wang. 2024. Dialectical Alignment: Resolving the Tension of 3H and Security Threats of LLMs. *ArXiv* abs/2404.00486 (2024).

[358] Wenkai Yang, Xiaohan Bi, Yankai Lin, Sishuo Chen, Jie Zhou, and Xu Sun. 2024. Watch Out for Your Agents! Investigating Backdoor Threats to LLM-Based Agents. *ArXiv* abs/2402.11208 (2024).

[359] Xianjun Yang, Xiao Wang, Qi Zhang, Linda Ruth Petzold, William Yang Wang, Xun Zhao, and Dahua Lin. 2023. Shadow Alignment: The Ease of Subverting Safely-Aligned Language Models. *ArXiv* abs/2310.02949 (2023).

[360] 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).

[361] Jia-Yu Yao, Kun-Peng Ning, Zhen-Hui Liu, Munan Ning, and Li Yuan. 2023. LLM Lies: Hallucinations are not Bugs, but Features as Adversarial Examples. *ArXiv* abs/2310.01469 (2023).

[362] 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.

[363] Jingwei Yi, Yueqi Xie, Bin Zhu, Keegan Hines, Emre Kiciman, Guangzhong Sun, Xing Xie, and Fangzhao Wu. 2023. Benchmarking and Defending against Indirect Prompt Injection Attacks on Large Language Models. *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.1* (2023).

[364] Yagmur Yigit, William J. Buchanan, Madjid G Tehrani, and Leandros A. Maglaras. 2024. Review of Generative AI Methods in Cybersecurity. *ArXiv* abs/2403.08701 (2024).

[365] Daniel Wankit Yip, Aysan Esmradi, and Chun Fai Chan. 2023. A Novel Evaluation Framework for Assessing Resilience Against Prompt Injection Attacks in Large Language Models. *2023 IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE)* (2023), 1–5.

[366] Wencong You, Zayd Hammoudeh, and Daniel Lowd. 2023. Large Language Models Are Better Adversaries: Exploring Generative Clean-Label Backdoor Attacks Against Text Classifiers. *ArXiv* abs/2310.18603 (2023).

[367] Jiahao Yu, Xingwei Lin, Zheng Yu, and Xinyu Xing. 2023. GPTFUZZER: Red Teaming Large Language Models with Auto-Generated Jailbreak Prompts. *ArXiv* abs/2309.10253 (2023).

[368] 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.

[369] Xiaodong Yu, Hao Cheng, Xiaodong Liu, Dan Roth, and Jianfeng Gao. 2023. Automatic Hallucination Assessment for Aligned Large Language Models via Transferable Adversarial Attacks. *ArXiv* abs/2310.12516 (2023).

[370] Tongxin Yuan, Zhiwei He, Lingzhong Dong, Yiming Wang, Ruijie Zhao, Tian Xia, Lizhen Xu, Binglin Zhou, Fangqi Li, Zhuosheng Zhang, Rui Wang, and Gongshen Liu. 2024. R-Judge: Benchmarking Safety Risk Awareness for LLM Agents. In *Conference on Empirical Methods in Natural Language Processing*.

[371] Zenghui Yuan, Yixin Liu, Kai Zhang, Pan Zhou, and Lichao Sun. 2023. Backdoor attacks to pre-trained unified foundation models. *arXiv preprint arXiv:2302.09360* (2023).

[372] 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).

[373] Yi Zeng, Hongpeng Lin, Jingwen Zhang, Diyi Yang, Ruoxi Jia, and Weiyan Shi. 2024. How Johnny Can Persuade LLMs to Jailbreak Them: Rethinking Persuasion to Challenge AI Safety by Humanizing LLMs. *ArXiv* abs/2401.06373 (2024).

[374] Yifan Zeng, Yiran Wu, Xiao Zhang, Huazheng Wang, and Qingyun Wu. 2024. AutoDefense: Multi-Agent LLM Defense against Jailbreak Attacks. *ArXiv* abs/2403.04783 (2024).

[375] Chong Zhang, Mingyu Jin, Qinkai Yu, Chengzhi Liu, Haochen Xue, and Xiaobo Jin. 2024. Goal-Guided Generative Prompt Injection Attack on Large Language Models. *2024 IEEE International Conference on Data Mining (ICDM)* (2024), 941–946.

[376] Chi Zhang, Zifan Wang, Ruoshi Zhao, Ravi Mangal, Matt Fredrikson, Limin Jia, and Corina S. Păsăreanu. 2024. Attacks and Defenses for Large Language Models on Coding Tasks. *2024 39th IEEE/ACM International Conference on Automated Software Engineering (ASE)* (2024), 2268–2272.

[377] Hangfan Zhang, Zhimeng Guo, Huaisheng Zhu, Bochuan Cao, Lu Lin, Jinyuan Jia, Jinghui Chen, and Di Wu. 2023. On the Safety of Open-Sourced Large Language Models: Does Alignment Really Prevent Them From Being Misused? *ArXiv* abs/2310.01581 (2023).

[378] Jinghuai Zhang, Jianfeng Chi, Zheng Li, Kunlin Cai, Yang Zhang, and Yuan Tian. 2024. Badmerging: Backdoor attacks against model merging. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*. 4450–4464.

[379] Junan Zhang, Kaifeng Huang, Yiheng Huang, Bihuan Chen, Ruisi Wang, Chong Wang, and Xin Peng. 2025. Killing Two Birds with One Stone: Malicious Package Detection in NPM and PyPI using a Single Model of Malicious Behavior Sequence. *ACM Transactions on Software Engineering and Methodology* 34, 4 (2025), 1–28.

[380] Jinghao Zhang, Yuting Liu, Qiang Liu, Shu Wu, Guibing Guo, and Liang Wang. 2024. Stealthy Attack on Large Language Model based Recommendation. *ArXiv* abs/2402.14836 (2024).

[381] Kaike Zhang, Qi Cao, Yunfan Wu, Fei Sun, Huawei Shen, and Xueqi Cheng. 2024. LoRec: Large Language Model for Robust Sequential Recommendation against Poisoning Attacks. *ArXiv* abs/2401.17723 (2024).

[382] Ruisi Zhang, Shehzeen Samarah Hussain, Paarth Neekhara, and Farinaz Koushanfar. 2023. REMARK-LLM: A Robust and Efficient Watermarking Framework for Generative Large Language Models. *ArXiv* abs/2310.12362 (2023).

[383] Rui Zhang, Hongwei Li, Rui Wen, Wenbo Jiang, Yuan Zhang, Michael Backes, Yun Shen, and Yang Zhang. 2024. Rapid Adoption, Hidden Risks: The Dual Impact of Large Language Model Customization. *ArXiv* abs/2402.09179 (2024).

[384] Xiaoyu Zhang, Cen Zhang, Tianlin Li, Yihao Huang, Xiaojun Jia, Xiaofei Xie, Yang Liu, and Chao Shen. 2023. A Mutation-Based Method for Multi-Modal Jailbreaking Attack Detection. *ArXiv* abs/2312.10766 (2023).

[385] Yuyang Zhang, Kangjie Chen, Xudong Jiang, Yuxiang Sun, Run Wang, and Lina Wang. 2024. Towards Action Hijacking of Large Language Model-based Agent. *arXiv preprint arXiv:2412.10807* (2024).

[386] Yuqi Zhang, Liang Ding, Lefei Zhang, and Dacheng Tao. 2024. Intention Analysis Prompting Makes Large Language Models A Good Jailbreak Defender. *ArXiv* abs/2401.06561 (2024).

[387] Yiming Zhang and Daphne Ippolito. 2023. Prompts should not be seen as secrets: Systematically measuring prompt extraction attack success. *arXiv preprint arXiv:2307.06865* 16 (2023).

[388] 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.

[389] Zhuo Zhang, Guangyu Shen, Guanhong Tao, Siyuan Cheng, and Xiangyu Zhang. 2023. Make Them Spill the Beans! Coercive Knowledge Extraction from (Production) LLMs. *ArXiv* abs/2312.04782 (2023).

[390] 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.

[391] 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).

[392] Jujia Zhao, Wenjie Wang, Chen Xu, Zhaochun Ren, See-Kiong Ng, and Tat-Seng Chua. 2024. Llm-based federated recommendation. *CoRR* (2024).

[393] Shuai Zhao, Meihuizi Jia, Anh Tuan Luu, Fengjun Pan, and Jinming Wen. 2024. Universal Vulnerabilities in Large Language Models: Backdoor Attacks for In-context Learning. In *Conference on Empirical Methods in Natural Language Processing*.

[394] Shuai Zhao, Jinming Wen, Anh Tuan Luu, Junbo Jake Zhao, and Jie Fu. 2023. Prompt as Triggers for Backdoor Attack: Examining the Vulnerability in Language Models. *ArXiv* abs/2305.01219 (2023).

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

[396] Fei Zheng. 2023. Input Reconstruction Attack against Vertical Federated Large Language Models. *ArXiv* abs/2311.07585 (2023).

[397] Tong Zhou, Yukui Luo, Shaolei Ren, and Xiaolin Xu. 2023. NNSplitter: An Active Defense Solution to DNN Model via Automated Weight Obfuscation. In *International Conference on Machine Learning*.

[398] Yukai Zhou and Wenjie Wang. 2024. Don't Say No: Jailbreaking LLM by Suppressing Refusal. In *Annual Meeting of the Association for Computational Linguistics*.

[399] Yifan Zhu, Huaibing Peng, Anmin Fu, Wei Yang, Hua Ma, Said F Al-Sarawi, Derek Abbott, and Yansong Gao. 2024. Towards robustness evaluation of backdoor defense on quantized deep learning models. *Expert Systems with Applications* 255 (2024), 124599.

[400] 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).

[401] 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).

[402] zscaler. 2024. *DeepSeek Lure Using CAPTCHAs To Spread Malware.* Retrieved April 20, 2024 from https://www.zscaler.com/blogs/security-research/deepseek-lure-using-captchas-spread-malware