SURVEY **Open Access**

When LLMs meet cybersecurity: a systematic literature review

Jie Zhang[1,2], Haoyu Bu[1,2], Hui Wen[1,2*], Yongji Liu[1,2], Haiqiang Fei[1,2], Rongrong Xi[1,2], Lun Li[1,2], Yun Yang[1,2], Hongsong Zhu[1,2*] and Dan Meng[1,2]

Abstract
The rapid development of large language models (LLMs) has opened new avenues across various fields, including cybersecurity, which faces an evolving threat landscape and demand for innovative technologies. Despite initial explorations into the application of LLMs in cybersecurity, there is a lack of a comprehensive overview of this research area. This paper addresses this gap by providing a systematic literature review, covering the analysis of over 300 works, encompassing 25 LLMs and more than 10 downstream scenarios. Our comprehensive overview addresses three key research questions: the construction of cybersecurity-oriented LLMs, the application of LLMs to various cybersecurity tasks, the challenges and further research in this area. This study aims to shed light on the extensive potential of LLMs in enhancing cybersecurity practices and serve as a valuable resource for applying LLMs in this field. We also maintain and regularly update a list of practical guides on LLMs for cybersecurity at https://github.com/tmylla/Awesome-LLM4Cybersecurity.

Keywords Cybersecurity, Cyber attack, Cyber defense, Large language model, Agent

Introduction

Large language models (LLMs), represented by advanced models such as ChatGPT (Ouyang et al. 2022), Llama (Touvron et al. 2023a), and their derivatives (Chiang et al. 2023; Almazrouei et al. 2023; Jiang et al. 2024a) have marked a significant advancement in artificial intelligence. By leveraging massive data and advanced neural network architectures, these models have demonstrated remarkable capabilities in understanding and generating human language (Zoph et al. 2022; Minaee et al. 2024). They not only set new benchmarks for achieving artificial general intelligence (AGI) but also show unique adaptability and effectiveness when collaborating with domain experts (Ge et al. 2024; Kaur et al. 2024). Such research enables LLMs to be tailored to specific challenges in various fields, thereby promoting progress and development in areas such as healthcare, law, education, and software engineering (Hou et al. 2023; Lai et al. 2024; Zhou et al. 2024b; Yan et al. 2024b; Li et al. 2023f; Zhao et al. 2024e). In the cybersecurity domain, exploring LLM applications can lay the foundations for further model development and utilization while highlighting potential transformative impacts (Yao et al. 2024b; Das et al. 2024; de Jesus Coelho da Silva and Westphall 2024; Motlagh et al. 2024; Yigit et al. 2024).

Cybersecurity is a critical issue given the growing number of cyber threats that pose significant risks to individuals, organizations, and governments (Thakur et al. 2015; Scala et al. 2019; Ghelani 2022). The rapid evolution and dynamic nature of cybersecurity poses challenges as adversaries continuously adapt strategies to exploit vulnerabilities and evade detection (Li and Liu 2021; Aslan et al. 2023). While traditional

*Correspondence:
Hui Wen
wenhui@iie.ac.cn
Hongsong Zhu
zhuhongsong@iie.ac.cn
[1] Institute of Information Engineering, Chinese Academy of Sciences, No. 19, Shucun Road, Haidian District, Beijing 100085, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China

approaches (e.g., signature-based detection, and rule-based systems) often struggle to keep pace with the evolving threat landscape, advancements in AI, particularly LLMs have opened new avenues for enhancing cybersecurity (Ferrag et al. 2024a). On one hand, open-sourced LLMs (e.g., LLaMA (Touvron et al. 2023a, b)) support the development of cybersecurity-enhanced domain LLMs such as RepairLlama (Silva et al. 2023) and Hackmentor (Zhang et al. 2023b) to address unique cybersecurity challenges. On the other hand, advanced LLMs such as ChatGPT solve complex tasks via prompt engineering, in-context learning, and chains-of-thought despite the lack of cybersecurity-specific training (Mohammed and Hossain 2024). These preliminary efforts show LLMs can aid cybersecurity tasks with promising results.

Despite the initial efforts of LLMs in cybersecurity, the field still faces several challenges (Das et al. 2024; Pankajakshan et al. 2024). First, many studies rely on case studies without comprehensive methodology, raising concerns about scalability and reproducibility. In addition, the field lacks connectivity and in-depth analysis between studies. With the rapid increase in the amount of LLM research in this field, conducting a systematic overview is essential to guide the field into a new stage of development, in which the application of LLM is not just experimental but also has a strategic impact (de Jesus Coelho da Silva and Westphall 2024; Motlagh et al. 2024; Yigit et al. 2024). Therefore, this work aims to conduct an extensive review of domain-specific LLMs tailored for cybersecurity, explore the breadth of LLM applications in this area, and identify emerging challenges to lay the foundation for future studies.

This survey aims to provide a comprehensive overview of the application of LLM in cybersecurity. We seek to address three key questions:

- RQ1: How to construct cybersecurity-oriented domain LLMs?
- RQ2: What are the potential applications of LLMs in cybersecurity?
- RQ3: What are the challenge and further research for the application of LLMs in cybersecurity?

By exploring these questions, we aim to bridge the gap between the advancement in LLMs and its potential impact on enhancing cybersecurity practices. We will delve into various cybersecurity tasks and applications to which LLMs are applicable, including vulnerability detection, secure code generation, program repair, binary, IT operations, threat intelligence, anomaly detection, and LLM-assisted attack, as shown in Table 1.

For the first question, we summarize the principles of existing cybersecurity LLMs, detailing their key techniques, the data used for model construction, and well-trained domain LLMs for special tasks. We provide insights into constructing domain models, which are valuable for researchers and practitioners looking to build customized LLMs based on specific requirements, such as computational limits, private data, and local knowledge bases (Sect. RQ1: How to construct cybersecurity-oriented domain LLMs?). For the second question, we conduct an extensive survey on the usage of existing LLMs in more than 10 cybersecurity tasks, including threat intelligence, vulnerability detection, program repairing, and others. This analysis not only helps us understand how LLMs benefit cybersecurity in various aspects but also allows us to identify their strengths when applied to domain-specific tasks. By demonstrating the diverse capabilities of LLMs, we aim to illustrate their potential to enhance and transform the cybersecurity field (Sect. RQ2: What are the potential applications of LLMs in cybersecurity?). The third question highlights the challenges that need to be overcome when applying LLMs in cybersecurity. LLMs' inherent vulnerabilities and susceptibilities lead to these attack challenges, especially attacks against LLMs and LLM jailbreaking. Additionally, we also explore further research directions for applying LLM to cybersecurity, guiding researchers and practitioners to promote advancement in this field (Sect. RQ3: What are the challenge and further research for the application of LLMs in cybersecurity?).

Table 1 The main cybersecurity tasks and applications where LLMs have been utilized

	Vulnerability detection	(In)secure code generation	Program repairing	Binary	IT operations	Threat intelligence	Anomaly detection	LLM assisted attack	Others
RQ1	✓	✓	✓	✓	✓	–	–	–	✓
RQ2	✓	✓	✓	–	–	✓	✓	✓	✓
RQ3	–	–	–	–	✓	–	✓	✓	–

In summary, this paper contributes by providing a comprehensive review of the state-of-the-art LLM applications in cybersecurity, highlighting the potential advantages and challenges, and proposing future research directions. The subsequent sections of this paper are organized as follows. Section Preliminaries outlines the scope of this paper. Section RQ1: How to construct cybersecurity-oriented domain LLMs? summarizes existing LLMs for cybersecurity. Section RQ2: What are the potential applications of LLMs in cybersecurity? details how LLMs can be applied to various cybersecurity tasks. Section RQ3: What are the challenge and further research for the application of LLMs in cybersecurity? highlights the challenges and promising opportunities for future research. Section Conclusion draws the conclusion.

Preliminaries

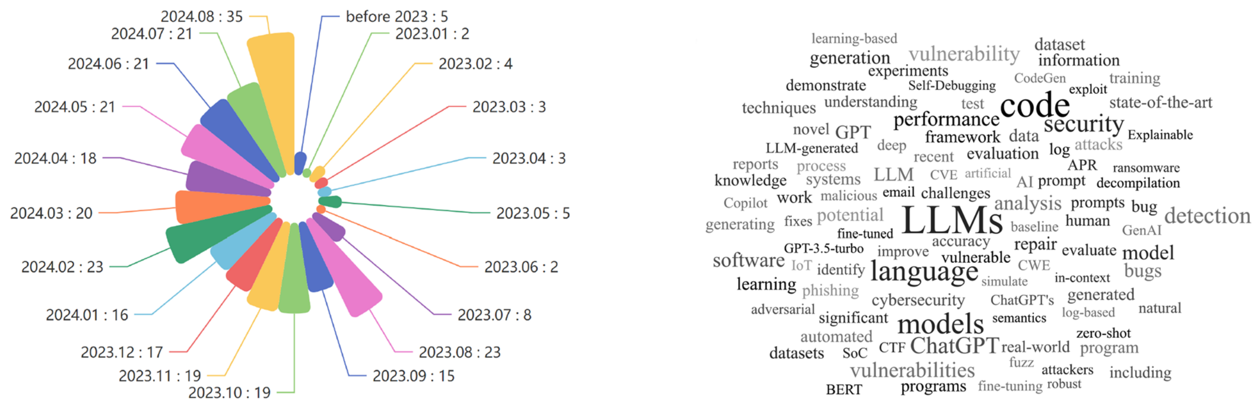
In this review paper, we systematically investigate the progress of LLMs' applications in cybersecurity, covering more than 300 academic papers since 2023. Through an exhaustive study and comprehensive analysis, we aim to provide a detailed overview of the current state, challenges, and future directions of LLM applications in cybersecurity. As shown in Fig. 1, this emerging research field continues to gain attention, and LLMs can be used to solve various tasks. This not only highlights the current and potential impact of LLMs in cybersecurity, but also offers practical guidance for future research. Therefore, this section first summarizes the surveyed papers from two aspects: one is the LLMs used in cybersecurity, and the other is the category of cybersecurity tasks to which LLMs can be applied.

LLMs in cybersecurity

LLMs have emerged as a transformative technology in the field of artificial intelligence, demonstrating remarkable capabilities in natural language understanding, generation, and reasoning (Brown et al. 2020; Zoph et al. 2022; Minaee et al. 2024). These models, trained with large amounts of data, have the potential to revolutionize various fields, including the critical area of cybersecurity (Motlagh et al. 2024; Yigit et al. 2024), as shown in Table 2. The application of LLMs in cybersecurity is expected to enhance threat detection, automated vulnerability analysis, intelligent defense mechanisms, and more.

LLMs can be categorized into two main types: open-source and closed-source models. Open-source LLMs (e.g., Llama (Touvron et al. 2023a) and Mixtral (Jiang et al. 2024a)) provide model weights, and researchers can fine-tune the models for specific cybersecurity tasks. This adaptability is particularly valuable in cybersecurity scenarios, such as private data and models fine-tuned to customized needs. However, open-source LLMs may lack the performance and scale of closed-source LLMs. On the other hand, closed-source LLMs (often referred to as commercial LLMs, e.g., ChatGPT (Ouyang et al. 2022) and Gemini (Team et al. 2023)), provide state-of-the-art performance and are maintained by commercial entities, often with access restrictions. While these models excel in accuracy and efficiency, their lack of transparency can raise concerns about potential biases and limitations in cybersecurity applications.

In the field of cybersecurity, there is a growing need for intelligent tools that can understand, analyze, and generate secure code. Code-based LLMs (e.g., CodeLlama (Roziere et al. 2023) and StarCoder (Li et al. 2023d; Lozhkov et al. 2024)) are particularly well suited to



(a) Time distribution over months (b) Word cloud

Fig. 1 Statistic of surveyed papers

Table 2 A Summary of LLMs used in cybersecurity (this paper)

Organization	LLMs	Size	Open-Source	Count	Link
OpenAI	GPT-3.5	175B	×	86	https://chat.openai.com/
	GPT-4	–	×	56	https://chat.openai.com/
	Codex	–	×	13	https://openai.com/blog/openai-codex
	davinci(-002,-003)	175B	×	9	https://openai.com/blog/openai-api
Google	Bard&Gemini	–	×	12	https://gemini.google.com/
	PaLM(-1,-2)	540B	×	7	https://ai.google.dev/models/palm
Anthropic	Claude(-1,-2)	–	×	2	https://claude.ai/
Github	Copilot	–	×	2	https://github.com/features/copilot
Microsoft	BingChat	–	×	2	https://www.bing.com/chat
EleutherAI	GPT-J	6B	✓	2	https://huggingface.co/EleutherAI/gpt-j-6b
	GPT-Neo	2.7B	✓	3	https://huggingface.co/EleutherAI/gpt-neo-2.7B
Meta	Llama(-1,-2)	7B/13B/70B	✓	38	https://huggingface.co/meta-llama
	LlamaGuard	7B	✓	1	https://huggingface.co/meta-llama/LlamaGuard-7b
	InCoder	1B/6B	✓	4	https://huggingface.co/facebook/incoder-1B
LMSYS	Vicuna	7B/13B	✓	12	https://huggingface.co/lmsys/vicuna-7b-v1.5
LianjiaTech	BELLE	7B/13B	✓	1	https://github.com/LianjiaTech/BELLE/
Databricks	Dolly	6B	✓	3	https://huggingface.co/databricks/dolly-v1-6b
–	Guanaco	7B	✓	2	https://huggingface.co/JosephusCheung/Guanaco
Salesforce	CodeGen(-1,-2)	3B/7B/16B	✓	9	https://github.com/salesforce/CodeGen/
	CodeT5	6B	✓	3	https://huggingface.co/Salesforce/codet5p-6b
BigCode	StarCoder(-1,-2)	3B/7B/15B	✓	3	https://huggingface.co/bigcode/
THUDM	ChatGLM	6B	✓	8	https://github.com/THUDM/ChatGLM-6B
KaistAI	Prometheus	7B/13B	✓	1	https://github.com/kaistAI/Prometheus
MistralAI	Mistral	7B	✓	6	https://huggingface.co/mistralai/Mistral-7B-v0.1
	Mixtral	8*7B	✓	5	https://huggingface.co/mistralai/Mixtral-8x7B-v0.1

address this demand. Unlike text-based LLMs that are trained on vast amounts of natural language data, code-based LLMs are specifically designed to understand and work with programming languages. Code-based LLMs are trained on large code bases covering multiple programming languages, allowing them to capture the complexity of syntax, semantics, and common coding patterns. This specialized training enables them to perform a variety of tasks, including code completion, bug detection, and automated code review. In the context of cybersecurity, these capabilities are useful for identifying potential vulnerabilities, suggesting secure coding practices, and remediating security vulnerabilities.

Cybersecurity categories of LLMs application

Cybersecurity has become a critical concern due to the increasing reliance on interconnected systems and the continued emergence of sophisticated cyber threats (Thakur et al. 2015; Ghelani 2022). The field of cybersecurity encompasses a wide range of practices, technologies, and strategies aimed at protecting computer systems, networks, and data from unauthorized access, attacks, damage, or disruption (Li and Liu 2021; Aslan et al. 2023). AI techniques, especially LLMs, have shown great potential in revolutionizing various aspects of cybersecurity (Yigit et al. 2024). The applications of LLMs in cybersecurity are wide-ranging, including threat intelligence, vulnerability detection, malware detection, and anomaly detection, fuzz and program repair, LLM assisted attack(in)secure code generation, and others.

- Threat Intelligence: It is very difficult to extract information from a large number of threat intelligence documents. Some researchers turn to LLMs to organize and analyze these massive and cluttered data.
- Vulnerability Detection: This is a critical task in cybersecurity, and has seen novel approaches emerge through the integration of LLMs.
- Malware Detection: LLMs can serve as both the static analysis assistant and the dynamic debugging assistant, improving the efficiency and effectiveness of the process.
- Anomaly Detection: It mainly refers to security anomalies such as malicious traffic in the flow, virus files in the system, anomalies in logs, etc.

- Fuzz: Traditional fuzzing techniques are effective in discovering software vulnerabilities, but their inherent limitations can affect their efficiency and effectiveness. The LLM-based approach for fuzzing is a promising area of research.
- Program Repair: Program repair is task-intensive and patching defects requires sufficient experience and knowledge. Many studies have proved the effectiveness of LLMs about this issue.
- LLM-Assisted Attacks: Many are not satisfied with LLMs' positive applications. They have discovered the effectiveness of LLMs in launching network attacks such as phishing emails and penetration testing.
- (In)secure Code Generation: Is there a risk in the code generated by LLMs? Moreover, can LLMs correct their code through some strategies?
- Others: In addition to the aspects mentioned above, we have also collected some researches which prove the importance of LLMs in the field of cybersecurity, there are fewer application studies of LLM in its field.

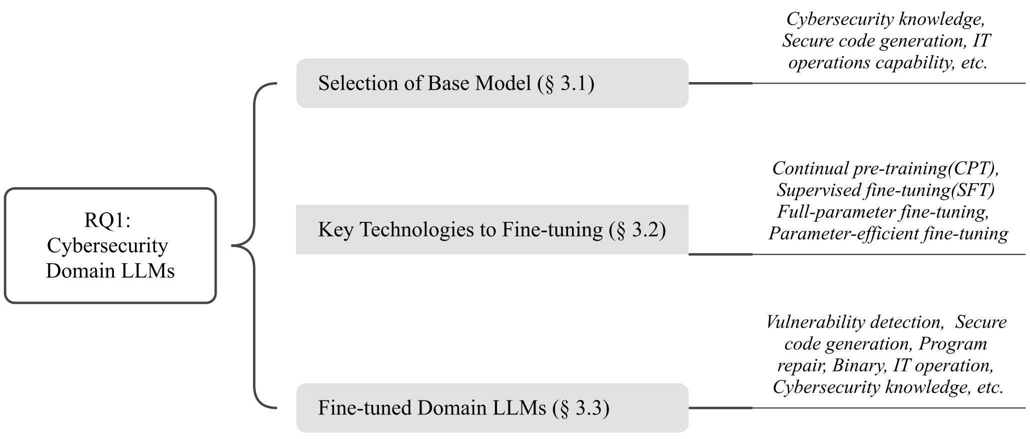
RQ1: How to construct cybersecurity-oriented domain LLMs?

The cybersecurity domain is facing escalating threats, demanding intelligent and efficient solutions to address complex and evolving attacks (Kaur et al. 2023; Kumar et al. 2023; Mijwil et al. 2023). LLMs provide new opportunities for the cybersecurity community (de Jesus Coelho da Silva and Westphall 2024; Motlagh et al. 2024). Trained on massive data, LLMs have acquired rich knowledge and developed strong understanding and reasoning capabilities, providing powerful decision-making for cybersecurity.

Advancing cybersecurity requires LLMs tailored to the field, leveraging their potential to learn domain-specific data and knowledge. This section firstly introduces several domain datasets for evaluating the cybersecurity capabilities of LLMs (Tihanyi et al. 2024b; Bhatt et al. 2023; Tony et al. 2023), while can guide the selection of an appropriate LLM as the base model when constructing cybersecurity LLMs. Then, we focus on key technologies for constructing cybersecurity LLMs, including training methods such as continual pre-training (CPT) (Çağatay Yıldız et al. 2024; Zhang et al. 2024e) and supervised fine-tuning (SFT) (Zhang et al. 2023c; Dong et al. 2023) of LLMs, as well as technical implementations like full-parameters fine-tuning and parameter-efficient fine-tuning (PEFT) (Ding et al. 2023). Finally, we summarize existing customized LLMs for specific cybersecurity tasks (Ferrag et al. 2023; Zhang et al. 2023b), including vulnerability detection, program repair, secure code generation, etc (Fig. 2).

Selection of base model for constructing domain LLM by evaluating cybersecurity capabilities

It is challenging to train a cybersecurity LLM from scratch. The general practice is to choose a general-purpose LLM as the base model and then fine-tune it. However, how do we select the appropriate base model among various LLMs? The basic idea is to choose the LLM with strong cybersecurity capabilities or those that perform well in specific security tasks. Such models are better at understanding and addressing security-related problems. Existing evaluation of LLM cybersecurity capabilities can be divided into several categories: cybersecurity knowledge, secure code generation, IT operations, Capture-the-Flag (CTF), and cyber intelligence, as listed in Table 3.



RQ1: Cybersecurity Domain LLMs
Selection of Base Model (§ 3.1)
Cybersecurity knowledge, Secure code generation, IT operations capability, etc.
Key Technologies to Fine-tuning (§ 3.2)
Continual pre-training(CPT), Supervised fine-tuning(SFT) Full-parameter fine-tuning, Parameter-efficient fine-tuning
Fine-tuned Domain LLMs (§ 3.3)
Vulnerability detection, Secure code generation, Program repair, Binary, IT operation, Cybersecurity knowledge, etc.

Fig. 2 An overview of RQ1

Table 3 A summary of datasets used for evaluating LLMs' cybersecurity capabilities

Catagory	Name	Count	Link
Cybersecurity knowledge	CyberBench	60000+	https://github.com/jpmorganchase/CyberBench
	CyberMetric	10000	https://github.com/cybermetric/CyberMetric
	SecEval	2000+	https://github.com/XuanwuAI/SecEval/
	SecQA	242	https://huggingface.co/datasets/zefang-liu/secqa
	SECURE	3602	https://github.com/aiforsec/SECURE
Secure code generation	CyberSecEval	7000+	https://github.com/facebookresearch/PurpleLlama
	LLMSeceval	150	https://github.com/tuhh-softsec/LLMSecEval/
	SecurityEval	121	https://github.com/s2e-lab/SecurityEval
	DegugBench	4253	https://github.com/thunlp/DebugBench
	PythonSecurityEval	470	https://github.com/Kamel773/LLM-code-refine
	Eyeballvul	24,000+	https://github.com/timothee-chauvin/eyeballvul
IT operations	NetEval	5732	https://huggingface.co/datasets/NASP/neteval-exam
	OpsEval	8920	https://github.com/NetManAIOps/OpsEval-Datasets
CTF	NYU CTF Dataset	200	https://github.com/NYU-LLM-CTF/NYU_CTF_Bench
	Cybench	40	https://github.com/andyzorigin/cybench/
Cyber threat intelligence	SEvenLLM	90000+	https://github.com/CSJianYang/SEevenLLM
	CTIBench	2500	https://github.com/xashru/cti-bench

Cybersecurity knowledge evaluation focuses on evaluating the model's understanding of cybersecurity concepts and its ability to provide accurate information on security threats and mitigation strategies. CyberBench (Liu et al. 2024f) is a domain-specific, multi-task benchmarking tool for evaluating LLMs' capabilities in cybersecurity tasks. It offers a generic and consistent approach that alleviates the limitations previously encountered in evaluating LLMs in this domain. SecEval (Li et al. 2023a) is designed to evaluate cybersecurity knowledge in LLMs. It provides more than 2000 multiple-choice questions in 9 domains: *Software Security, Application Security, System Security, Web Security, Cryptography, Memory Safety, Network Security, and PenTest*. By facilitating the evaluation of ten state-of-the-art foundational models, this study provides new insights into their performance in the cybersecurity domain. By combining expert knowledge with the collaboration of LLMs, Tihanyi et al. (2024b) create the CyberMetric benchmark dataset, which contains 10,000 questions and is designed to evaluate the cybersecurity knowledge of various LLMs within the cybersecurity field. SecQA (Liu 2023) is a dataset of multiple-choice questions generated by GPT-4 based on the textbook "Computer Systems Security: Planning for Success," which is designed specifically to assess LLMs' understanding and application of security principles. SecQA provides questions at two tiers of complexity, which can not only serve as an assessment tool but also facilitate the advancement of LLM applications in environments that require a high level of security awareness. In addition, SECURE (Bhusal et al. 2024) is a benchmark designed to assess LLMs' performance in realistic cybersecurity scenarios, which includes 6 datasets to evaluate the capabilities of knowledge extraction, understanding, and reasoning in the Industrial Control System scenarios.

Secure code generation tests the model's capability to generate code that is not only functional but also adheres to security best practices, aiming to minimize vulnerabilities. CyberSecEval (Bhatt et al. 2023) is a security coding benchmark that aims at assessing the potential security risks and tendencies to facilitate cyber attacks when LLMs generate code. By evaluating seven models including Llama 2, Code Llama, and OpenAI's GPT, CyberSecEval effectively pinpoints key cybersecurity risks and provides practical insights for model improvement. LLMSecEval (Tony et al. 2023) is a dataset of 150 natural language prompts based on the narrative descriptions of various vulnerabilities that appear in MITRE's Top 25 Common Weakness Enumeration (CWE) rankings. LLMSecEval evaluates the security of LLM-generated code by comparing it to secure implementation examples for each prompt. SecurityEval (Siddiq and Santos 2022) focuses on the security evaluation of code generation models to prevent the creation of vulnerable code and thus avoid potential misuse by developers. This dataset includes 130 samples covering 75 types of vulnerabilities mapped to CWE. PythonSecurityEval (Alrashedy and Aljasser 2024) is a real-world dataset collected from actual scenarios on Stack Overflow, which is designed to evaluate LLMs' ability to generate secure Python code and their capacity to fix security vulnerabilities. DebugBench (Tian et al. 2024) has 4,253 instances covering four major bug categories and 18 minor types in C++, Java, and Python. This comprehensive evaluation clarifies the advantages and disadvantages of LLMs in automated debugging, which marks a major step in understanding their applicability and restraint in practical coding scenarios. EvilInstructCoder (Hossen et al. 2024) is designed to assess the cybersecurity vulnerabilities of instruction-tuned Code LLMs to adversarial attacks. By incorporating practical threat models to reflect real-world adversaries with varying capabilities and evaluating the exploitability of instruction-tuned Code LLMs under these diverse adversarial attack scenarios. Eyeballvul (Chauvin 2024) is a benchmark designed to test the vulnerability detection capabilities of language models at scale, which have contained 24,000+ vulnerabilities across 6,000+ revisions and 5,000+ repositories.

IT operations capability is used to evaluate the model's proficiency in managing and securing IT infrastructures, including awareness of security situations, security threat analysis, and incident response. NetEval (Miao et al. 2023) is an evaluation set designed to measure the common knowledge and reasoning abilities of LLMs in NetOps. This set contains 5,732 questions related to NetOps, covering five different NetOps subdomains. With NetEval, researchers systematically evaluate the NetOps capabilities of 26 publicly available LLMs. Additionally, OpsEval (Liu et al. 2024b) contains 7184 multi-choice questions and 1736 question-answering formats in English and Chinese. It aims to analyze the root cause of faults, operational script generation, and alert information summarization to evaluate the performance of LLMs in IT operational tasks comprehensively. Donadel et al. (2024) develop a thorough framework for evaluating LLMs' capabilities in various network-related tasks and conduct an exhaustive study on LLMs' comprehension of computer networks.

In addition, NYU CTF Dataset (Shao et al. 2024) and Cybench (Zhang et al. 2024a) are used to assess LLMs capacity to solve Capture the Flag (CTF) challenges in cybersecurity, aiming to improve the efficiency of LLMs in interactive cybersecurity tasks and automated task planning. SEvenLLM (Ji et al. 2024) is a framework to benchmark, elicit, and improve cybersecurity incident analysis and response abilities in LLMs for security events. CTIBench (Alam et al. 2024) is a benchmark designed to evaluate the performance of LLMs in Cyber Threat Intelligence (CTI) applications, encompassing multiple datasets focused on assessing the knowledge acquired by LLMs within the cyber-threat landscape.

The evaluation of LLMs' cybersecurity capabilities not only guides the basic model during fine-tuning but also demonstrates that general LLMs have certain cybersecurity capabilities. This supports the feasibility of directly using LLMs (without fine-tuning) to aid cybersecurity applications, as discussed in section 4. Furthermore, these studies help researchers and developers recognize the limitations of LLMs in the field of cybersecurity, thereby providing the direction for artificial intelligence toward higher standards and more professional security development.

Key technologies in constructing domain LLMs

LLMs have demonstrated remarkable language understanding and generation capabilities by leveraging the transformer architecture and self-supervised pre-training strategies (Vaswani et al. 2017; Radford and Narasimhan 2018; Brown et al. 2020). However, developing a specialized LLM for cybersecurity from scratch requires a lot of computational resources, which is impractical for most research teams. Fortunately, existing general LLMs have acquired extensive knowledge and demonstrated excellent generalization capabilities (Touvron et al. 2023a, b; Yang et al. 2023; Jiang et al. 2024a). By combining these pre-trained LLMs with domain datasets for training, we can adopt a more efficient approach to enhance the model's cybersecurity capabilities. This approach not only significantly reduces the computational demands of pre-training, but also maximizes the use of the knowledge that LLMs have learned. Thereby, the model can understand and perform cybersecurity-related tasks, such as automated threat detection, vulnerability identification, and security policy recommendations.

To apply general LLMs to cybersecurity, researchers mainly employ two approaches: continual pre-training (CPT) and supervised fine-tuning (SFT).

Continual pre-training involves further training of pre-trained LLMs using a large amount of unlabeled domain-specific data (Çağatay Yıldız et al. 2024; Zhang et al. 2024e; Wu et al. 2024c; Ibrahim et al. 2024). This method aims to improve the model's understanding and application of domain knowledge, significantly improving its broad applicability within the cybersecurity field. CPT is based on the core assumption that even after extensive pre-training, the model still has the potential for further enhancement, especially in specific domains or tasks. The process usually involves several key steps: first, select a dataset that can appropriately represent the characteristics of the target domain; second, determine the strategy for continuous pre-training; and finally, perform pre-training and adjust the model architecture or optimization algorithm as needed to adapt to the new training objectives.

Supervised Fine-Tuning uses labeled domain-specific data for training, enabling direct optimization of the model's performance on specific cybersecurity tasks (Zhang et al. 2023c; Dong et al. 2023). Compared to CPT, SFT focuses on improving the performance of a specific task. In SFT, the model weights are refined via gradients calculated from a task-specific loss function. This function quantifies the deviation between the model's predictions and the actual labels, thus promoting the learning of task-oriented patterns. SFT relies on the utilization of high-quality, human-annotated data, which is a collection of prompts and their corresponding responses. SFT is important for LLMs such as ChatGPT, which are designed to follow user instructions and focus on specific tasks in context. This specific type of fine-tuning is also referred to as instruction fine-tuning.

In the context of CPT and SFT, researchers have the option of employing either full-parameter fine-tuning (FULL) or parameter-efficient fine-tuning (PEFT).

Full-parameter fine-tuning is a classical approach that adjusts all parameters of the model during training. This allows the model to fully adapt and specialize to the target domain. By optimizing all parameters, the model can achieve optimal performance for specific tasks or datasets. However, full parameter updates require considerable computing power and time, posing challenges in efficiency and scalability, especially as the number of LLM parameters continues to increase.

Conversely, PEFT methods fine-tune only a small number of model parameters or additional parameters while freezing most parameters of the pre-trained LLMs, which greatly reduces the computational costs. It also helps in portability, and users can fine-tune the model using PEFT methods to obtain tiny checkpoints of only a few MB in size. In summary, PEFT methods are favored because they enable users to obtain comparable performance to full fine-tuning while having only a small number of trainable parameters. There are several PEFT methods, such as adapter tuning, prefix tuning, prompt tuning, LoRA, QLoRA, and so on:

Adapter tuning (He et al. 2021) inserts adapters after the multi-head attention and feed-forward layers in the transformer architecture, which updates only the parameters in the adapter during fine-tuning while keeping the rest of the model parameters frozen. *P-tuning* (Liu et al. 2023e) automatically learns optimal task-specific prompt embeddings by introducing trainable prompt tokens, eliminating the need for manual prompt design and potentially improving performance with the addition of anchor tokens. *Prefix tuning* (Liu et al. 2021) keeps the language model parameters frozen and optimizes small, continuous, task-specific vectors called prefixes. *Prompt tuning* (Lester et al. 2021) fine-tunes for specific tasks through learning soft prompts by backpropagating and merging labeled examples. *LoRA* (Hu et al. 2022) is a small trainable submodule that can be inserted into the transformer architecture. It freezes the pre-trained model weights and inserts a trainable low-rank decomposition matrix into each layer of the model, reducing the number of trainable parameters for downstream tasks. After training, the matrix parameters are combined with the original LLM. *QLoRA* (Dettmers et al. 2024) is a further optimization of LoRA, which carries out gradient backpropagation to a low-rank adapter with a frozen 4-bit quantized pre-trained language model, reducing the memory requirement for fine-tuning while being almost comparable to full fine-tuning.

By integrating these techniques, researchers can select appropriate methods to construct LLMs tailored to the specific needs of the cybersecurity domain, as shown in Fig. 3. Furthermore, emerging technologies also provide insights for the construction of cybersecurity LLMs. For example, model editing techniques (Yao et al. 2023b; Zhang et al. 2024c) can precisely modify LLMs to incorporate cybersecurity knowledge without negatively affecting other knowledge. Prompt engineering (Bozkurt and Sharma 2023; Ye et al. 2023; Sahoo et al. 2024), by designing effective prompts to guide LLMs towards desired outputs, can alleviate the bottleneck of training data and resources required for constructing cybersecurity LLMs.

Fine-tuned domain LLMs for cybersecurity

The researchers have used the above techniques and base models to customize LLMs to address specific problems in the field of cybersecurity, as shown in Table 4. These efforts highlight the great potential of integrating domain-specific knowledge to enhance the capabilities of LLMs, especially for key applications including vulnerability detection, fault Localization, program repair, and so on.

Vulnerability detection involves identifying and classifying potential security vulnerabilities in software code. Shestov et al. (2024) fine-tunes WizardCoder (Luo et al. 2024) with LoRA specifically for vulnerability detection, focusing on the binary classification of whether Java functions contain vulnerabilities. Ferrag et al. (2023) performs partial parameters fine-tuning on FalconLLM (Almazrouei et al. 2023) using C code samples to obtain SecureFalcon, which can distinguish between vulnerable and non-vulnerable samples with a detection accuracy of up to 96%, and further proposes a method for repairing vulnerabilities using FalconLLM. Yang et al. (2024a) introduces a new fault localization method based on the language model, named LLMAO. LLMAO adds bidirectional adapter layers on CodeGen (Nijkamp et al. 2023b,

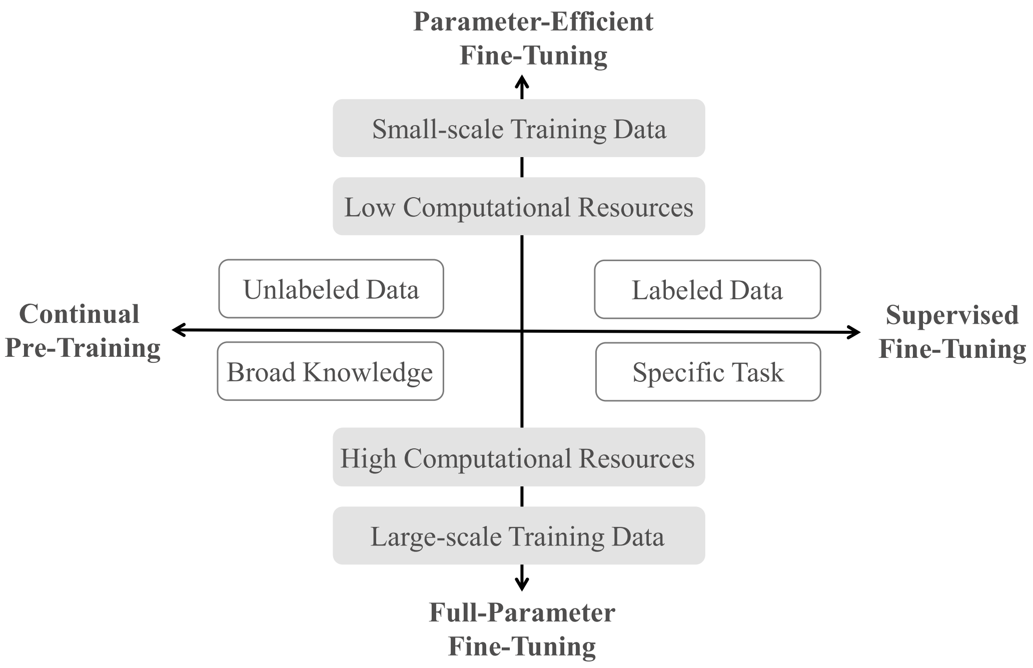

Parameter-Efficient Fine-Tuning
Small-scale Training Data
Low Computational Resources
Unlabeled Data
Labeled Data
Continual Pre-Training
Supervised Fine-Tuning
Broad Knowledge
Specific Task
High Computational Resources
Large-scale Training Data
Full-Parameter Fine-Tuning

Fig. 3 Comparison of domain LLM training approaches. *Continual Pre-Training* and *Supervised Fine-Tuning* offer methods to enhance domain-specific performance based on existing LLMs, while *Full Parameter Fine-Tuning* and *Parameter-Efficient Fine-Tuning* represent different technical pathways within these training processes

Table 4 A summary of fine-tuned domain LLMs for cybersecurity

	Description	Base model	Training methods	Open source
SecureFalcon	Software vulnerability classification	Falcon	SFT(PEFT)	No
VulLLM	Vulnerability detection	StarCoder	SFT(PEFT)	No
SC-GPT	Vulnerability constraint decoding	GPT-J	SFT(FULL)	No
RepairLLaMA	Automatic program repair	Llama	SFT(PEFT)	Yes
LLMAO	Fault location	CodeGen	SFT(PEFT)	No
OWL	IT operation	Llama	SFT(PEFT)	No
SafeCoder	Secure code generation	Llama	SFT(PEFT)	No
HackMentor	Security knowledge QA	Llama	SFT(PEFT)	Yes
IoT-LM	Internet of Things	Llama	CPT+SFT(PEFT)	No
Nove	Binary code analysis	DeepSeek-Coder	CPT+SFT(FULL)	Yes

a), enabling the model to learn bidirectional representations of codes and predict the probability of defects in code lines. Detect Llama (Ince et al. 2024) is fine-tuned on Code-Llama with 17k dataset, outperforming GPT-4 in smart contract vulnerability detection.

Secure code generate via LLMs aims to improve the security of automatically generated code by mitigating vulnerability risks. Storhaug et al. (2023) proposes a new approach called vulnerability-constrained decoding, which integrates vulnerability tags during model training. By avoiding generating code with these labels, the model significantly reduces the generation of vulnerable code. Fine-tuning on GPT-J (Wang 2021) shows a notable reduction in vulnerabilities in the generated code. He et al. (2024) focuses on improving the security of code generation by LLMs via instruction tuning. They convert CodeLlama (Roziere et al. 2023) to SafeCoder using supervised fine-tuning on a dataset containing both secure and insecure programs. This approach achieves significant security improvements (approximately 30%) across various popular LLMs and datasets while remaining practical.

Automated program repair aims to automatically fix software bugs without human intervention. Silva et al. (2023) proposes a new program repair approach called RepairLLaMA, which significantly improves LLMs' program repair capabilities by applying LoRA fine-tuning to CodeLlama. It outperforms GPT-4 on the Java benchmarks Defects4J and HumanEval-Java. Li et al. (2024a) first creates an instruction dataset APR-INSTRUCTION by using prompt engineering, then fine-tunes LLMs using four different PEFT methods based on this data to improve the model's automated program repair capabilities.

Binary is the most basic form of computer code, it is important to learn what it means and how to use it. Jiang et al. (2023a) demonstrates the benefits of LLMs for binary analysis. They continually train StarCoder (Li et al. 2023d; Lozhkov et al. 2024) on specialized binary code corpus and new tasks, leading to the development of Nova and Nova$^+$. After SFT, the enhanced LLMs effectively address specific tasks such as binary code similarity detection, binary code translation, and binary code recovery.

IT operations manage routine tasks and activities to keep the infrastructure running for other services. Guo et al. (2024a) describes a specialized LLM for IT operations, named Owl, which is supervised fine-tuned of Llama on the collected Owl-Instruct dataset. Owl outperforms existing models in IT-related tasks and demonstrates effective generalization capabilities on the Owl-Bench benchmark.

Cybersecurity knowledge assistants help to improve users' security awareness and assist users in defending against cyber attacks through interaction with users. Zhang et al. (2023b) proposes Hackmentor, a cybersecurity knowledge assistant. They develop a dataset of cybersecurity instructions and conversations and train Hackmentor using LoRA by fine-tuning on Llama and Vicuna (Chiang et al. 2023). CyberPal (Levi et al. 2024) is fine-tuned using SecKnowledge, a domain knowledge-driven cybersecurity instruction dataset, to build a security-specialized LLM capable of answering and following complex security-related instructions. This demonstrates the potential of LLMs in cybersecurity applications.

In addition to enhancing the cybersecurity capabilities of general LLMs through SFT and CPT, specialized security-oriented LLMs can be developed by leveraging innovative model architectures and proprietary large-scale datasets for independent pretraining. The Machine Language Model (MLM) is a large model designed for the machine language domain, utilizing an innovative architecture to align multimodal data across machine language, natural language, and source code (Liu et al. 2018; Wang et al. 2022, 2024b). This approach not only addresses the limitations of existing LLMs in comprehending machine language but also introduces transformative advancements in software reverse engineering and software security detection. TrafficFormer (Zhou et al. 2024a) is an efficient pre-training model designed for traffic data. Given the characteristics of traffic data, it introduces fine-grained multi-classification tasks in the pre-training stage to enhance the representation of traffic data; in the fine-tuning stage, it uses the random initialization characteristics of the field to propose a traffic data enhancement method to help the traffic model focus on key information. In this way, the accuracy of the model's traffic detection and protocol understanding is improved. These developments pave the way for novel research directions in the field of cybersecurity.

Answer to Q1: For researchers, it is feasible to construct the domain LLM by fine-tuning a general LLM with cybersecurity data using methods such as CPT and SFT, and the implementation technique depends on the specific application scenario, resource availability, and the expected level of performance improvement.

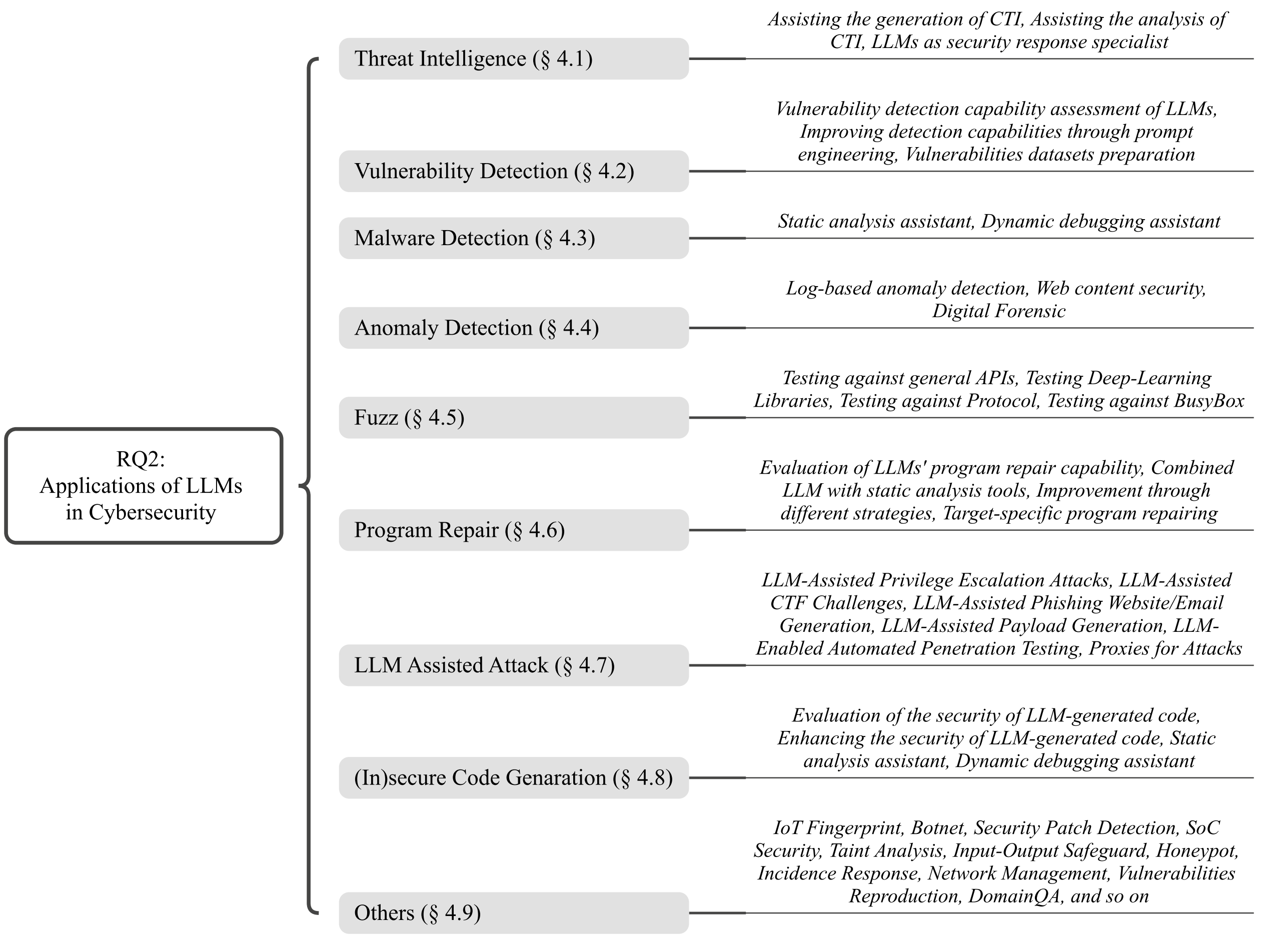

RQ2: Applications of LLMs in Cybersecurity
Threat Intelligence (§ 4.1)
Assisting the generation of CTI, Assisting the analysis of CTI, LLMs as security response specialist
Vulnerability Detection (§ 4.2)
Vulnerability detection capability assessment of LLMs, Improving detection capabilities through prompt engineering, Vulnerabilities datasets preparation
Malware Detection (§ 4.3)
Static analysis assistant, Dynamic debugging assistant
Anomaly Detection (§ 4.4)
Log-based anomaly detection, Web content security, Digital Forensic
Fuzz (§ 4.5)
Testing against general APIs, Testing Deep-Learning Libraries, Testing against Protocol, Testing against BusyBox
Program Repair (§ 4.6)
Evaluation of LLMs' program repair capability, Combined LLM with static analysis tools, Improvement through different strategies, Target-specific program repairing
LLM Assisted Attack (§ 4.7)
LLM-Assisted Privilege Escalation Attacks, LLM-Assisted CTF Challenges, LLM-Assisted Phishing Website/Email Generation, LLM-Assisted Payload Generation, LLM-Enabled Automated Penetration Testing, Proxies for Attacks
(In)secure Code Genaration (§ 4.8)
Evaluation of the security of LLM-generated code, Enhancing the security of LLM-generated code, Static analysis assistant, Dynamic debugging assistant
Others (§ 4.9)
IoT Fingerprint, Botnet, Security Patch Detection, SoC Security, Taint Analysis, Input-Output Safeguard, Honeypot, Incidence Response, Network Management, Vulnerabilities Reproduction, DomainQA, and so on

Fig. 4 An overview of RQ2

Table 5 LLMs for cyber threat intelligent

Category	Related work
CTI generation (5)	Mitra et al. (2024), Perrina et al. (2023), Fayyazi and Yang (2023), Schwartz et al. (2024), Michelet and Breitinger (2024)
CTI information extraction (8)	Clairoux-Trepanier et al. (2024), Siracusano et al. (2023), Hu et al. (2024), Zhang et al. (2024g), Fieblinger et al. (2024), Wu et al. (2024e), Fayyazi et al. (2024), Singla et al. (2023)
CTI report deduplication (1)	Zhang et al. (2023d)
LLMs as security response experts (5)	Lin et al. (2023), Kaheh et al. (2023), Jin et al. (2024), Tseng et al. (2024a), Rajapaksha et al. (2024)

RQ2: What are the potential applications of LLMs in cybersecurity?

This section introduces the application of LLMs in various cybersecurity tasks, encompassing offline defense (e.g., threat intelligence), online defense (e.g., vulnerability detection, malware detection, and anomaly detection), software testing (e.g., fuzz and program repair), attack assistance (e.g., LLM assisted attack), source code generation and analysis (e.g., (in)secure code generation), and other security-related applications (e.g., honeypot, botnet, SoC security, etc.). By reviewing the key advancements in each topic, this paper aims to offer a comprehensive perspective on the evolution of the cybersecurity landscape driven by LLMs integration (Fig. 4).

Threat intelligence

Since LLMs have shown excellent analysis and summarization capabilities in natural language processing tasks, it is natural for LLMs to be used to process threat intelligence text. For example, Clairoux-Trepanier et al. (2024)

assesses the performance of an LLM system built on the GPT to extract CTI information, highlight the relevance of using LLMs for CTI. Researchers have used LLMs to assist in the generation and analysis of cyber threat intelligence (CTI), as shown in Table 5.

Mitra et al. (2024) introduces a framework known as LocalIntel, which aims to provide users with reliable threat intelligence by allowing LLMs to summarize knowledge after querying global and local knowledge databases. Global knowledge mainly refers to well-documented reports on cybersecurity threats from CWE and CVE, while local knowledge is customized by the organization for practical purposes to supplement global knowledge. Perrina et al. (2023) also conducts similar work to extract security knowledge from a wide range of knowledge bases and automatically generate reports using LLMs. A few similar efforts are as follows. Fayyazi and Yang (2023) employs LLM to generate descriptions of cyber attacks and fine-tune the model using information collected from ATT&CK and CAPEC. Then, they compare the performance of the fine-tuned LLMs with the directly used LLMs (GPT—3.5) in describing attacks. LMCloudHunter (Schwartz et al. 2024) leverages LLMs to automatically generate generic signature detection rule candidates from textual and visual OSCTI data.

Specifically for digital forensics, Michelet and Breitinger (2024) proposes a method to automate the generation of reports. They study the structure of forensic reports to identify common sections and assess the feasibility of LLMs in generating these sections. Through a case study approach, the article evaluates the strengths and limitations of LLMs in creating different sections of forensic reports.

Given that most threat intelligence providers offer information in an unstructured format, Siracusano et al. (2023) and Hu et al. (2024) propose innovative solutions to the common problem of extracting useful information from threat intelligence. The former designs a framework named aCTIon, which includes downloading and parsing raw reports, extracting useful information with LLM, and exporting structured reports following STIX (Barnum 2012) standard. The latter constructs the knowledge graph of unstructured threat intelligence and fine-tunes LLMs to automate information extraction tasks. Also, by leveraging the capabilities of LLMs in instruction prompting and in-context learning, Zhang et al. (2024g) propose a fully automatic LLM-based framework, AttacKG, which comprises four consecutive modules: rewriter, parser, identifier, and summarizer, to construct attack knowledge graphs from CTI. Fieblinger et al. (2024) explore the application of open-source LLMs for extracting meaningful triples from CTI texts. Then, the extracted data is utilized to construct a knowledge graph, offering a structured and queryable representation of threat intelligence. Besides, considering the quality assessment of threat intelligence provided by intelligence platforms, Wu et al. (2024e) propose a novel CTI quality assessment framework that combines knowledge graphs and LLMs. In this verifier, LLMs automatically extract OSCTI key claims to be verified and utilize a knowledge graph consisting of paragraphs for fact-checking. This significantly improves the performance of LLMs in intelligence quality assessment.

In another work, Fayyazi et al. (2024) studies the application of LLMs in cybersecurity to explain and summarize cyberattack Tactics, Techniques, and Procedures (TTPs) from the MITRE ATT&CK framework. It reveals that RAG significantly improves the explanation of TTPs by providing relevant context, highlighting the potential of LLMs in threat intelligence. Singla et al. (2023) discusses the capability of LLMs to automatically analyze and summarize software supply chain security vulnerabilities. Their results show that LLMs show good potential, especially when the data is comprehensive, but still cannot replace human analysts in this specific field.

In addition to extracting valuable information from large amounts of text, report deduplication is also an important research focus in this field. Zhang et al. (2023d) uses LLMs to alleviate the problem of bug report deduplication. They leverage LLMs as an intermediate step to improve the performance of REP (Sun et al. 2011) (a traditional method of measuring the similarity between bug reports) by identifying keywords, thereby improving its effectiveness.

There are also studies that attempt to use LLMs as experienced security response experts. Lin et al. (2023) uses LLMs as suggestion providers to mitigate vulnerabilities through prompt engineering. They design a system that is able to retrieve relative CVE & CWE information after the user enters a vulnerability description. LLMs' mitigation suggestions are a subcomponent of the system. Kaheh et al. (2023) believes that LLMs are not only question-answering assistants with expertise but also able to perform actions based on the user's description (e.g., instructing the host's intrusion detection system to block a specific IP). To enhance strategic reasoning in cybersecurity, Jin et al. (2024) introduces Crimson, a system that uses LLMs to associate CVEs with MITRE ATT&CK techniques to improve threat prediction and defense. The core concept is the Retrieval-Aware Training (RAT) process, which refines LLMs to generate accurate cybersecurity strategies, thereby significantly reducing errors and hallucinations. By integrating real-time data retrieval and domain-specific fine-tuning, Crimson enhances the models' interpretability and strategic consistency, providing a proactive approach to cybersecurity threat intelligence.

Table 6 LLMs for vulnerability detection

Category	Related work
Vulnerability detection capability assessment (14)	Zhou et al. (2024c), Tamberg and Bahsi (2024), Zhou et al. (2024d), Mahyari (2024), Mao et al. (2024a), Cheshkov et al. (2023), Purba et al. (2023), Omar and Shiaeles (2023), Khare et al. (2023), Jensen et al. (2024), Shestov et al. (2024), Li et al. (2023c), Kouliaridis et al. (2024), Guo et al. (2024b)
Vulnerability detection capability improvement (18)	Wang et al. (2023b), Bakhshandeh et al. (2023), Mathews et al. (2024), Zhang et al. (2024b), Lee et al. (2024), Yang et al. (2025), Wang et al. (2024g), Yang et al. (2024e), Du et al. (2024b), Hu et al. (2023b), Liu et al. (2023g), Sun et al. (2024b), Du et al. (2024a), Sun et al. (2024a), Mao et al. (2024b), Li et al. (2024d), Chen et al. (2023a), Liu et al. (2023c)
Vulnerability detection datasets preparation (4)	Chen et al. (2023b), Gao et al. (2023), Tihanyi et al. (2023), Gonçalves et al. (2024)

Tseng et al. (2024a) develop an AI agent designed to automate the labor-intensive and repetitive tasks associated with analyzing CTI reports. By leveraging the advanced capabilities of LLMs, the AI agent can accurately extract important information from large volumes of text and generate Regex to help SOC analysts accelerate the process of establishing correlation rules. Rajapaksha et al. (2024) introduces a QA model based on Retrieval Augmented Generation (RAG) techniques together with LLMs and provides answers to the users' queries based on the knowledge base that contains curated information about cyber-attacks investigations and attribution or on outside resources provided by the users.

Vulnerability detection

This section provides an overview of the main studies on vulnerability detection using LLMs (we blur the concepts of "vulnerability" and "software defect" in this part). Through these studies, we aim to shed light on the progress, challenges, and future directions of leveraging LLMs to enhance cyber security (Table 6).

Whether LLMs have the ability to detect vulnerabilities? The following papers conduct preliminary studies on this question. Although their results may vary due to some unknown reasons (e.g., they may use different datasets), in general, they all show that LLMs are promising for vulnerability detection (Zhou et al. 2024c; Tamberg and Bahsi 2024; Zhou et al. 2024d; Mahyari 2024; Mao et al. 2024a).

Cheshkov et al. (2023) initially evaluates whether GPT-3 and GPT—3.5 could identify some known CWE vulnerabilities in Java code. The results show that the application effect in vulnerability detection tasks is not good and needs further improvement and research. In another work, Purba et al. (2023) uses LLMs (including GPT—3.5, CodeGen, and GPT-4) to analyze several common vulnerabilities (e.g., SQL injection, overflow). The conclusion confirms that LLMs do have the ability to detect vulnerabilities, but the false positive rate is high. However, Omar and Shiaeles (2023) fine-tunes GPT on various vulnerable code benchmarks to detect software vulnerabilities and achieve good performances. Similarly, Khare et al. (2023) concludes that LLMs are generally able to perform better vulnerability detection than existing static analysis and deep learning-based tools. With carefully designed prompts, desirable results can be obtained on synthetic datasets, but performance degrades on more challenging real-world datasets. Jensen et al. (2024) compares the performance of a wide range of open-source and proprietary models with Python code snippets in assisting vulnerability discovery. Their research suggests that LLMs can be effectively used to enhance the efficiency and quality of code reviews, particularly in detecting security issues within software code. Shestov et al. (2024) fine-tunes WizardCoder for vulnerability detection and investigate whether the encountered performance limit is due to the limited capacity of CodeBERT-like models. Their results suggest that this is indeed the case and that LLMs have great potential for application in vulnerability detection. Li et al. (2023c) presents LLift, a framework that leverages LLMs to assist static program analysis, specifically for detecting use-before-initialization (UBI) defects. LLift interacts with static analysis tools and LLMs, demonstrating 50% accuracy in real-world scenarios and identifying 13 previously unknown UBI bugs in the Linux kernel. Kouliaridis et al. (2024) assess the ability of various LLMs to detect Android code vulnerabilities listed in the latest Open Worldwide Application Security Project (OWASP) Mobile Top 10. While the reported findings regarding code vulnerability analysis show promise, they also reveal significant discrepancies among the different LLMs. Moreover, Guo et al. (2024b) thoroughly analyzes the capabilities of LLMs in detecting vulnerabilities within source code by testing the models beyond their usual applications. It also paves the way for LLM-based vulnerability detection by addressing two key aspects: model training and dataset curation

Improving detection capabilities through different strategies. Instead of directly providing code to LLM and asking it to answer, many researchers would adopt various strategies in advance. They believe that simply providing code is not enough and that the code needs to be further preprocessed or more information needs to be provided to LLMs for vulnerability reasoning.

Wang et al. (2023b) proposes a code sequence embedding (CSE) that combines the AST, DFG, and CFG of the code as input to the model. Then, the model captures the semantic information with the help of conformer mechanism (Gulati et al. 2020), an improved architecture of Transformer. Zhang et al. (2024b) not only provides the code to GPT but also provides the API call sequence and data flow diagrams. Bakhshandeh et al. (2023) conducts a similar experiment to compare the performance of the model when different levels of information are given, including asking for the vulnerability point directly, giving some CWE information, and telling LLMs what vulnerabilities are in the code. Mathews et al. (2024) focuses on Android platform vulnerabilities and compares the performance of LLMs on three conditions: asking LLMs to find vulnerabilities directly, providing vulnerability summaries before asking and granting LLMs permission to request any file in the APK after providing the APK core (AndroidManifest.xml and MainActivity.java). Lee et al. (2024) focuses on the security of Android systems against filesystem vulnerabilities. They present PathSentinel, which leverages LLMs to generate targeted exploit code based on the identified vulnerabilities and generated input payloads, reducing the engineering effort required for writing test applications. DLAP (Yang et al. 2025) combines the advantages of deep learning models for specific tasks and LLM's powerful general understanding ability, and achieves excellent vulnerability detection performance. Wang et al. (2024g) reframes vulnerability detection as an anomaly detection task by viewing vulnerable code as an anomaly within the LLM's predicted code distribution. This approach frees the model from the need for labeled data, allowing it to learn a representation of vulnerable code. Ultimately, it results in a detector that identifies software vulnerabilities at the line-level granularity.

There are some studies that use retrieval-augmented generation (RAG) based on additional knowledge bases to facilitate LLM for vulnerability detection. Yang et al. (2024e) explores three different strategies for augmenting both single and multi-statement vulnerabilities using LLMs: Mutation, Injection, and Extension. These strategies potentially alleviate the shortage of data. Du et al (2024b) proposed Vul-RAG, which leverages knowledge-level RAG framework to detect vulnerability. And the vulnerability knowledge generated by Vul-RAG can serve as high-quality explanations to further improve the manual detection accuracy.

In addition to the above efforts, researchers have also proposed many innovative ideas to improve the vulnerability detection ability of LLMs. Hu et al. (2023b) proposes an innovative two-stage framework named GPTLENS, which includes two adversarial agent roles: auditor and critic. The auditor performs during the generation phase and its main goal is to identify potential vulnerabilities in the smart contract. In contrast, the critic works during the identification phase its main goal is to evaluate the vulnerabilities generated by the auditor. Liu et al. (2023g) uses traditional algorithms (TF-IDF and BM25) to match the code under analysis with the code in the vulnerability corpus to determine similarity. The code under analysis is presented to LLMs together with similar corpus entries. Based on in-context learning, LLMs can more accurately determine whether the code contains the identified vulnerability type. Sun et al. (2024b) specifically focuses on vulnerability detection in smart contracts and introduce a tool called GPTScan. GPTScan first parses the smart contract project to determine the reachability of the functions, retaining only those that may have vulnerabilities. Subsequently, GPTScan uses GPT to match candidate functions with predefined vulnerability types. Finally, GPTScan asks GPT to confirm the vulnerability. VulLLM (Du et al. 2024a) combines multi-task learning with LLMs, introducing two auxiliary tasks-vulnerability localization and vulnerability explanation-in addition to the primary vulnerability detection task. This approach enhances the model's ability to understand the root causes of code vulnerabilities, thereby improving its generalization capabilities.

To improve LLM's ability to reason about vulnerabilities, Sun et al. (2024a) proposes LLM4Vuln, which separates the vulnerability reasoning capabilities of LLMs from others (e.g., proactively seeking more information, employing relevant vulnerability knowledge, and following instructions to output structured results). They allow LLMs to request additional contextual information about the target code. Moreover, they conclude that the more information input to LLMs is not the better. Too much information such as full vulnerabilities report, and a large amount of invocation context, may lead to distractions. Mao et al. (2024b) proposes a new method called MuCoLD, which simulates a multi-role code review process for vulnerability detection in software. By playing different roles, such as developers and testers, LLMs participate in discussions to reach a consensus on the existence and classification of vulnerabilities. IRIS (Li et al. 2024d) combines LLMs with static analysis to enable reasoning over the entire codebase. It automatically infers taint specifications and

Table 7 LLMs for malware detection

Category	Related work
Static analysis assistant (10)	Pearce et al. (2022b), Tan et al. (2024), Fang et al. (2024a),Zhao et al. (2023), Palacio et al. (2023), Yan et al. (2023), Fujima et al. (2023), Wang (2023), Zahan et al. (2024), Lu et al. (2024)
Dynamic debugging assistant (3)	Tian et al. (2024), Liu et al. (2024e), Ahmad et al. (2023a)

performs contextual analysis, thereby reducing reliance on human-generated specifications and manual inspection.

In addition to detecting vulnerabilities in specific programs, recent studies have attempted to use LLMs to infer lists of affected libraries from vulnerability reports. Chen et al. (2023a) observes that many vulnerability reports in the national vulnerability database (NVD) either omitted affected libraries or provided incomplete or incorrect library names, increasing the risk of third-party library vulnerabilities. To address this problem, they propose VulLibGen, a method designed to detect vulnerabilities in third-party libraries. VulLibGen takes only vulnerability descriptions as input and uses the inherent knowledge of LLMs to generate a list of library names that may be affected by the reported vulnerabilities.

Liu et al. (2023c) explores the application of ChatGPT for vulnerability management. They evaluate ChatGPT's capabilities in predicting security bugs, evaluating severity, repairing vulnerabilities, and verifying patch correctness. The results reveal that while ChatGPT can assist in identifying and mitigating software security threats, it needs enhancements to perform more nuanced tasks, such as vulnerability prioritization and patch validation.

Construction of vulnerability detection datasets. In addition to the methods of retraining or fine-tuning the models, the construction of the dataset is also important for vulnerability detection.

Chen et al. (2023b) introduces a new vulnerable source code dataset called DiverseVul, which contains 18,945 vulnerable functions (covering 150 CWEs) and 330,492 non-vulnerable functions, all written in C/C++. They also explore 11 different deep learning architectures and conclude that despite the remarkable success of LLMs, they still face challenges such as high false positive rates, low F1 scores, and difficulty in identifying complex CWEs for vulnerability detection. Gao et al. (2023) introduces a comprehensive vulnerability benchmark dataset called VulBench, which includes high-quality data from CTF challenges and real-world applications with detailed annotations of vulnerability types and causes for each vulnerable function. Tihanyi et al. (2023) creates a dataset containing 112,000 vulnerable C code instances with detailed information about the specific vulnerability, including CWE number, location, and function name. Notably, all the code in this dataset is generated by GPT—3.5, which illustrates the application potential of vulnerable code synthesized by LLMs. Source Code Processing Engine (SCoPE) (Gonçalves et al. 2024) is a framework that incorporates strategized techniques to reduce the size and normalize C/C++ functions. Additionally, SCoPE refines the CVEFixes dataset, which can be used for fine-tuning pre-trained LLMs for software vulnerability detection.

Malware detection

In malware detection, LLMs can serve as both the static analysis assistant and the dynamic debugging assistant, improving the efficiency and effectiveness of the process, and making it an important part of defending against cyber threats (Table 7).

LLMs as the static analysis assistant

Pearce et al. (2022b) explores the application of LLMs, such as OpenAI's Codex, in the field of reverse engineering, particularly in understanding software functionality and extracting information from the code. LLMs are primarily used to analyze the functionality of C code provided by reverse engineering tools such as Ghidra. These C codes are obtained from binary files through

Table 8 LLMs for anomaly detection

Category	Related work
Log-based anomaly detection (6)	Karlsen et al. (2024), Liu et al. (2023b), Qi et al. (2023), Han et al. (2023), Liu et al. (2024c), Zhang et al. (2024f)
Web content security (9)	Jamal and Wimmer (2023), Wu et al. (2024d), Nahmias et al. (2024), Heiding et al. (2023), Vörös et al. (2023), Guastalla et al. (2023), Ferrag et al. (2024b), Ziems et al. (2023), Ali and Kostakos (2023)
Digital forensic (1)	Scanlon et al. (2023)

the process of decompilation. Decompilation is also an important task in reverse engineering. Tan et al. (2024) introduces an LLM tailored for decompilation that focuses on converting compiled machine code back into human-readable source code. They fine-tune a model called DeepSeek-Coder on a large number of C code and assembly code pairs and evaluate the performance of their work by recompiling and executing the decompiled code. Fang et al. (2024a) explores the potential and limitations of LLMs for code analysis tasks, especially when dealing with obfuscated code. In the experiments, they conduct tests that allow LLMs to generate de-obfuscated versions of code, i.e., to recover more readable original code from obfuscated code.

Zhao et al. (2023) focuses on how to improve LLM's semantic understanding of programs through fuzz testing. Their core idea is that programs with their basic units (e.g., functions, and subroutines) are designed to exhibit diverse behaviors and provide possible outputs given different inputs. Thus, through fuzz testing, various inputs trigger different functions of the code that can help LLMs understand the behavior and semantics of the program more thoroughly.. Palacio et al. (2023) introduces ASTxplainer, an explainability method for LLMs in coding scenarios. It aligns token predictions with Abstract Syntax Tree (AST) nodes, enabling detailed evaluation and visualization of model predictions. ASTxplainer consists of AsC-Eval for structural performance estimation, AsC-Causal for causal analysis, and AsC-Viz for visualization. These components provide a more comprehensive explanation of how LLMs work when generating or analyzing code.

Yan et al. (2023) focuses on how LLMs can be utilized to aid in dynamic analysis of malware. The core idea of the research is to use GPT-4 to generate explanatory text for each API call, and then use BERT to generate a series of API sequences to be executed based on the previous analysis. This approach can theoretically generate representations for all API calls without the need to train the dataset during the generation process. Fujima et al. (2023) uses LLM (specifically ChatGPT) to analyze the linguistic and strategic elements of ransomware communications. By examining a range of ransomware samples, the study identifies patterns and strategies used in ransom notes, revealing the evolution of ransomware strategies characterized by sophisticated language use and psychological manipulation. Wang (2023) also discusses the potential and challenges of LLMs in generating strategies against ransomware. Zahan et al. (2024) employs GPT-3 and GPT-4 to detect potential malware in the npm ecosystem by analyzing JavaScript packages. The study introduces SocketAI Scanner, a multi-stage workflow that utilizes iterative self-refinement, zero-shot role-playing, and chain of thought prompting techniques to enhance the model's ability to identify malicious intent within code. By comparing LLMs' performance with static analysis tools, the paper demonstrates that LLMs can effectively pinpoint malware with higher precision and lower false positive rates.

Binary malware summarization aims to automatically generate human-readable descriptions of malware behaviors from executable files, facilitating tasks like malware cracking and detection. Lu et al. (2024) introduces a novel code summarization framework, namely MALSIGHT, which can iteratively generate descriptions of binary malware by exploring malicious source code and benign pseudocode. At the same time, they construct the first malware summary dataset, MalS and MalP, to support further research.

LLMs as the dynamic debugging assistant

Tian et al. (2024) introduces DebugBench, a benchmark for evaluating LLMs' debugging capabilities in programming. It consists of 4253 instances across various bug categories in C++, Java, and Python. The benchmark is constructed by collecting code snippets from LeetCode, implanting bugs with GPT-4, and conducting rigorous quality assessment. Liu et al. (2024e) addresses the challenge of automated Graphical User Interface (GUI) testing for mobile applications. They propose a novel approach called GPTDroid that formulates the GUI testing as a question and answering (Q&A) task, where the LLM is asked to chat with the mobile apps by passing GUI page information to generate testing scripts. These scripts are executed and iterations of the application's responses are fed back to the model to guide further exploration. Ahmad et al. (2023a) proposes an approach called FLAG to assist human debuggers in identifying and localizing security and functional bugs in code. FLAG takes a code file as input and regenerates each line in the file for comparison. It compares the original code with LLM-generated code to flag notable differences as anomalies for further inspection.

Anomaly detection

We investigate some methods to incorporate LLMs into cybersecurity frameworks for anomaly detection, underscoring their critical role in maintaining network integrity and safeguarding against cyber intrusions (Table 8).

Log-based anomaly detection

Karlsen et al. (2024) tests 60 language models fine-tuned for log analysis, including models with different architectures such as BERT, RoBERTa, DistilRoBERTa, GPT-2 and GPT-Neo. The results show that these fine-tuned models can be effectively used for log analysis, especially for domain adaptation for specific log types. Targeting service logs on Huawei Cloud, Liu et al. (2023b) proposes a framework called ScaleAD, which aims to provide an accurate, lightweight, and adaptive solution for log anomaly detection in cloud systems. When ScaleAD's Trie-based Detection Agent (TDA) detects suspicious anomaly logs, it queries the LLM to validate these logs. The LLM determines whether the logs are anomalous or not by understanding the semantics of the log content and gives the corresponding confidence scores. Qi et al. (2023) proposes a log anomaly detection framework named LogGPT. This framework consists of three main components: log preprocessing, prompt construction, and response parser. The log preprocessing component filters, parses and groups raw log messages into a structured format for further analysis. The response parser extracts the output returned by ChatGPT for detailed analysis and evaluation of the detected anomalies. Han et al. (2023) performs similar work. The difference is that they fine-tune GPT-2 by introducing a Top-K reward metric, which directs the model to focus on the most relevant parts of the log sequence, thus improving the accuracy of anomaly detection. Liu et al. (2024c) introduces an online log analysis method called LogPrompt. They employ LLMs to parse unstructured logs and generate reports with a specific structure. LogPrompt then utilizes chain of thought and in-context learning methods to progressively reason about log content and provide normal/abnormal judgments. Zhang et al. (2024f) introduces LEMUR, a cutting-edge log parsing framework that enhances log analysis with entropy sampling for efficient log clustering and semantic understanding using LLMs. LEMUR addresses the limitations of traditional parsers by discarding manual rules and focusing on semantic information. Relying on semantic understanding of LLMs, the framework accurately distinguishes between parameters and invariant tokens, leading to impressive efficiency and state-of-the-art performance in log template merging and categorization.

Web content security

LLMs can assist in the detection of phishing and spam. Jamal and Wimmer (2023) presents a model named Improved Phishing and Spam Detection Model (IPSDM), a fine-tuned model based on DistilBERT and RoBERTa. They emphasizes the potential of LLMs to revolutionize the field of email security and suggests that these models can be valuable tools for improving the security of information systems. Another work also conduct spam detection with LLMs, Wu et al. (2024d) evaluate ChatGPT's performance in spam detection and find it outperforms BERT on a low-resource Chinese dataset but lags on a larger English dataset. The study also highlights the positive impact of increasing prompts on ChatGPT's accuracy. Nahmias et al. (2024) introduces a spear-phishing detection approach utilizing LLMs to generate "prompted contextual document vectors." By posing targeted questions to LLMs about email content, the method quantifies the presence of common persuasion principles, creating vectors that capture the malicious intent within spear-phishing emails. The approach utilizes the reasoning capabilities of LLMs and outperforms traditional phishing detection methods. In addition to detecting phishing emails, there are studies on generating phishing emails using LLMs. Heiding et al. (2023) evaluates the performance of GPT-4 in creating phishing emails and compare its effectiveness with traditional phishing methods called V-Triad method, which relys on manual design based on general rules and cognitive heuristics. They also explore the use of LLMs in detecting phishing emails, where models like GPT, Claude, PaLM, and LLaMA demonstrate strong capabilities in identifying malicious intent, sometimes surpassing human detection rates.

In addition, LLMs can be used for malicious URLs, DDoS attacks, and other cyber threat detection. Based on the website content, Vörös et al. (2023) uses the knowledge distillation approach to detect malicious URLs. Specifically, unlabeled URLs are classified and labels are generated by a teacher model. The student model trained with this label improves accuracy with significantly fewer parameters and is therefore suitable for malicious URL detection. Guastalla et al. (2023) explores the potential of LLMs in detecting DDoS attacks by investigating the performance of LLMs on two datasets. For the CICIDS 2017 dataset, they fine-tune LLMs with labeled pcap files to enable traffic classification through few-shot learning. Urban IoT dataset is a real-world anonymized dataset containing 4060 IoT devices. Considering the complexity of this dataset, they fine-tune LLMs separately depending on whether the correlation of traffic between IoT devices is considered or not. Ferrag et al. (2024b) encodes the network traffic by employing a novel encoding technique called Privacy-Preserving Fixed-Length Encoding (PPFLE). Then they train a model named SecurityBERT with these encoded data to perform a classification task on network traffic. Specifically, their model targets IoT devices to achieve efficient and accurate cyber threat detection on resource-limited IoT devices. Ziems et al. (2023) studies the interpretation of decision tree models in network intrusion detection (NID) systems. They convert the path and structure data of the decision tree into text format and provide it to LLMs to generate explanations. Moreover, LLMs provide additional background knowledge to help users understand why certain features are important in categorization. Ali and Kostakos (2023)

Table 9 LLMs for Fuzz

Category	Related work
Fuzz capability assessment (6)	Jiang et al. (2024b), Wang et al. (2024a), Zhang et al. (2023f), Hu et al. (2023a), Xia et al. (2024), Lemieux et al. (2023)
Testing against General APIs (1)	Zhang et al. (2023a)
Testing against DL Libraries (2)	Deng et al. (2023b), Deng et al. (2023c)
Testing against Protocol (1)	Meng et al. (2024)
Testing against BusyBox (1)	Asmita et al. (2024)

introduces HuntGPT, a system that integrates LLMs with traditional machine learning for anomaly detection. The system utilizes a random forest classifier trained on the KDD99 dataset to identify cyber threats. To enhance interpretability, the system employs XAI techniques such as SHAP and Lime and combines them with the GPT—3.5 conversational agent.

Digital forensic

Scanlon et al. (2023) assesses the applicability of ChatGPT for digital forensics. ChatGPT is used to help determine if a file has been downloaded to a PC and if the file has been executed by a specific user. In addition, ChatGPT is also used to detect browser history, Windows event logs, and interactions with cloud platform machines.

Fuzz

Although traditional fuzzing techniques are effective in discovering software vulnerabilities, their inherent limitations can affect their efficiency and effectiveness. One significant drawback is that traditional fuzzers operate in a largely random or semi-random manner, which is time-consuming and inefficient because they may not explore all possible execution paths. Additionally, the mutated seeds are usually artificially crafted, which makes the time and labor costs high. Although all of the above problems have been studied for many years and there are many ways to mitigate them, the emergence of LLMs provides a new way of thinking in the field of fuzz testing (Jiang et al. 2024b; Wang et al. 2024a), as shown in Table 9.

What are the advantages of LLMs fuzz over traditional methods?

Zhang et al. (2023f) evaluates the performance of ChatGPT in generating test cases directly (without tuning) and compare it with two traditional testing tools (i.e., SIEGE, and TRANSFER). Their experiments show that LLMs outperform traditional methods in generating test cases when a detailed description of the vulnerability, possible exploits, and code context are given.

There are some advantages of LLMs over traditional tools. One of the most important factors is that LLMs lead to a shift from random mutation to guided mutation. Hu et al. (2023a) introduces a GPT-based seed mutator to the traditional gray-box fuzz testing, selecting seeds from a seed pool and requesting variants from ChatGPT to generate higher-quality inputs. Another factor is that LLMs have a strong understanding of programming languages, enabling them to perform testing tasks in multiple languages. Most traditional methods can only fuzz specific programming languages. Xia et al. (2024) tests 6 languages code (i.e., C, C++, Go, SMT2, Java, and Python) with a method named Fuzz-Loop, which automatically mutates test cases based on LLMs. Most traditional fuzz methods fail to achieve high code coverage in all codes, while LLMs have mastered the logic of code and can generate more targeted test cases for areas with low coverage. For example, Lemieux et al. (2023) uses Codex to generate test cases against low-coverage functions when SBST (Search-Based Software Testing, a traditional fuzz method) reaches coverage plateau. Specifically, the raw character sequences generated by the Codex are deserialized into an internal test case representation compatible with SBST to leverage its mutation operations and fitness functions.

Specific fuzzing strategies for different testing objects

Depending on the test subject, the strategy should be adjusted when fuzzing with LLMs. For testing against general APIs, Zhang et al. (2023a) investigates the effectiveness of LLMs in generating invocation code. They compare LLM-based generation with traditional program analysis methods and find that LLMs can automatically generate a large number of effective fuzzing drivers while reducing human intervention. The research introduces query strategies, iterative improvements, and the use of examples to enhance LLM performance. Although it's all about testing APIs, the strategy for testing against deep-learning libraries needs to be modified. Because programs that call deep learning libraries usually have strict requirements on tensor dimensions, ignoring this would cause the fuzzer to perform meaningless tests.

Deng et al. (2023b) proposes TitanFuzz, a tool specifically for generating test cases for deep learning libraries. Their training corpus contains a large number of code snippets that call the DL library APIs, so that the language syntax and semantics, and complex DL API constraints can be learned to efficiently generate DL programs. FuzzGPT (Deng et al. 2023c) is also about fuzzing the DL library. The difference is that FuzzGPT focuses on using historical error-triggered code snippets to guide LLMs to generate test cases.

In addition to the above research, we have collected some studies targeting other testing objects. Testing against Protocol. Meng et al. (2024) discusses how to find security vulnerabilities in protocol implementations in the absence of a machine-readable protocol specification. They train LLMs with massive human-readable protocol documents and ask LLMs to mutate interactive messages for protocol fuzz (e.g., HTTP). Testing against BusyBox. Specifically targeting BusyBox, a popular utility in Linux-based devices, Asmita et al. (2024) introduces two fuzzing methods. One is to use LLMs to generate target-specific initial seeds for fuzzing, which significantly improves the efficiency of identifying crashes and potential vulnerabilities. The other is crash reuse, which employs previously acquired crash data to streamline the testing process for new targets.

Program repairing

The software development lifecycle is deeply impacted by the presence of bugs, with their detection and resolution being costly. Researchers are motivated to find new ways to automatically identify and correct bugs/vulnerabilities with LLMs (Zhang et al. 2024d), as shown in Table 10.

Evaluation of existing LLMs on program repairing

For state-of-the-art LLMs (open-sourced or proprietary), many studies have evaluated their capabilities for program repairing. Prenner and Robbes (2021) explores the application of OpenAI's Codex to the field of automatic program repair (APR), specifically its ability to locate and fix bugs in software. They use the QuixBugs benchmark, which includes 40 bugs in Python and Java, to evaluate the effectiveness of Codex in APR tasks. Notably, Codex outperforms numerous existing APR methods even without retraining. Sobania et al. (2023) conducts similar work with the previous one. Both studies evaluate LLMs for automatic program repair on QuixBugs benchmark. In this work, ChatGPT is evaluated instead of Codex. Keller and Nowakowski (2024) discusses the application of Gemini in automating the repair of software vulnerabilities, especially for vulnerabilities found by the sanitizer tool in C/C++, Java, and Go code. The authors argue that while the success rate seems low, it has the potential to significantly reduce engineering effort over time. Yu et al. (2024) evaluates the performance of three LLMs, Gemini Pro, GPT-4, and GPT—3.5, on codes with identified vulnerabilities from real-world code reviews. The findings indicate that GPT-4 outperforms the other models, but all LLMs have great potential, especially for conciseness, clarity, and accuracy of responses. Xia et al. (2022) selects 9 LLMs and compare them with traditional automated program repair methods, demonstrating the superior effectiveness of LLMs in this field.

Pearce et al. (2023) explores the potential of LLMs for zero-shot vulnerability repair in code. Through extensive experiments with various LLMs in synthetic, artifactual, and real-world security scenarios, they demonstrate that while LLMs show promise in repairing simple cases, they struggle with more complex, real-world examples. The study reveals the limitations and strengths of LLMs in cybersecurity and urges further research into the application of LLMs in program repairing. Wu et al. (2023) compares the capabilities of LLMs and deep learning-based APR models in fixing Java vulnerabilities. They evaluate the performance of 5 LLMs (Codex, CodeGen, CodeT5, PLBART, and InCoder), 4 fine-tuned LLMs, and 4 deep learning-based APR techniques on two real-world Java vulnerability benchmarks (i.e., Vul4J and VJBench). They design code transformations to address the overlapping of train and test sets faced by Codex, and create a new

Table 10 LLMs for program repairing

Category	Related work
Program repairing capability assessment (8)	Prenner and Robbes (2021), Sobania et al. (2023), Keller and Nowakowski (2024), Yu et al. (2024),Xia et al. (2022), Pearce et al. (2023), Wu et al. (2023), Xiang et al. (2024)
Program repairing capability improvement (16)	Xu et al. (2024a), Chen et al. (2024a), Ahmed and Devanbu (2023), Kulsum et al. (2024),Zhao et al. (2024a), Yang et al. (2024b), Yin et al. (2024),Wei et al. (2023), Islam et al. (2024), Wang et al. (2024h), Kong et al. (2024),Chen et al. (2024b), Yang et al. (2024c), de-Fitero-Dominguez et al. (2024), Zhao et al. (2024d),Dehghan et al. (2024)
Combined LLMs with static analysis tools (2)	Alrashedy and Aljasser (2024),Jin et al. (2023)
Target-specific program repairing (6)	Tol and Sunar (2023), Paria et al. (2023), Ahmad et al. (2023b), Charalambous et al. (2024), Le et al. (2024), Xu et al. (2024c)

Java vulnerability remediation benchmark, VJBench, to better evaluate LLMs and APR techniques. Xiang et al. (2024) investigate LLM-based function-level APR, focusing on the effects of the few-shot learning mechanism and the inclusion of auxiliary repair-relevant information. The study shows that LLMs with zero-shot learning are already effective for function-level APR, but applying the few-shot learning mechanism results in varying repair performance. Additionally, they find that directly incorporating auxiliary repair-relevant information into LLMs significantly enhances function-level repair performance.

Combined LLMs with static analysis tools

Instead of using LLMs alone for program repair, some studies have combined them with traditional program analysis tools to increase the efficiency of those tools. Alrashedy and Aljasser (2024) proposes a new approach called Feedback-Driven Security Patching (FDSP), which passes feedback from Bandit to LLM. With the help of the static code analysis tool, LLM can generate potential solutions to address security vulnerabilities. Each suggested solution, along with the corresponding vulnerable code segment, is fed back to LLM for verification and validation. Jin et al. (2023) introduces a program repair framework called InferFix that incorporates the latest static analyzers for fixing critical security and performance vulnerabilities. Inferfix consists of two main components: a retriever and a generator. The retriever aims to search for semantically similar vulnerabilities and their associated fixes. The generator is fine-tuned on vulnerability fix data, with prompts enhanced by bug type annotations and semantically similar fixes, thereby improving the model's ability to generate effective proposals.

Improving repair capabilities through different strategies

To improve the performance of LLMs on program repair tasks, researchers have proposed some methodologies. D4C (Xu et al. 2024a) is a straightforward prompting framework for APR. By aligning the output to LLMs training objective and allowing LLMs to refine the whole program without first identifying faulty statements, D4C greatly improve LLM's APR capability. Chen et al. (2024a) proposes an approach called SELF-DEBUGGING. Even if there is no human feedback about the correctness of the code or error messages, this method can identify the error by observing the execution results and explaining the code generated by natural language. Ahmed and Devanbu (2023) explores the application of Self-Consistency (an approach for improving model reasoning ability (Wang et al. 2023d)) in program repair. By incorporating commit-logs as reasoning paths in few-shot prompts, Self-Consistency enables LLMs to generate diverse solutions. The most frequent solution from multiple samples is selected to improve patch accuracy. Similarly, VRpilot (Kulsum et al/ 2024) is based on reasoning and patch validation feedback. The method uses a chain-of-thought prompt to reason about a vulnerability before generating patch candidates and iteratively refines the prompts based on feedback from external tools on previously generated patches, improving patch accuracy. DRCodePilot (Zhao et al. 2024a) is designed to enhance GPT-4-Turbo's APR capabilities by incorporating design rationales (DR) into the prompt instruction, along with a utility feedback-based self-reflective framework. This framework prompts GPT-4 to reconsider and refine its outputs by referencing the provided patch and suggested identifiers.

Additionally, Yang et al. (2024b) introduces a novel approach that leverages the entropy of LLMs in combination with prior APR tools to enhance all stages of the APR process. By using entropy-delta for patch ranking and classification, this method can rank correct patches more effectively than state-of-the-art machine learning tools. ThinkRepair (Yin et al. 2024) is an LLM-based autonomous two-stage automatic program repair framework. In the collection stage, CoT prompts guide the LLM to automatically gather various reasoning chains that form the foundation of the repair knowledge. In the repair stage, sample selection is performed for few-shot learning, with interactive feedback from the LLM. This approach significantly improves LLMs' bug fixing capability. Wei et al. (2023) proposes a program repair framework named Repilot. It starts by masking the buggy code segment and then utilizes LLMs to generate candidate patches. During the generation, Repilot consults the completion engine to prune infeasible tokens and proactively completes the code when necessary. This approach enhances the compilation rate and correctness of patches while reducing the number of invalid attempts in the generation process. Islam et al. (2024) introduces SecRepair, a system that uses LLMs to detect and fix code vulnerabilities in the software. It utilizes reinforcement learning with the semantic reward mechanism to improve the model's ability to generate accurate code comments and descriptions, guiding developers to address security issues. ARJA-CLM (Wang et al. 2024h) integrates a multi-objective evolutionary algorithm with a code language model to fix multi-location bugs in Java projects. It does this by predicting the correct statement for masked buggy positions using the powerful code-filling capabilities of CodeLLMs. Kong et al. (2024) launches Contrastrepair to provide more accurate feedback by providing LLMs with contrastive test case pairs (a failing test and a passing test), thereby enhancing conversation-driven repair framework. The key insight is to minimize the difference between the generated passing test and the failing one, effectively isolating bug causes. ContrastRepair interacts with ChatGPT repeatedly to generate patches until a plausible fix is generated. Unlike previous function-level approaches, Chen et al. (2024b) investigates the performance of LLMs in repository-level program repair, which needs to consider interactions and dependencies between code that may span multiple functions or files. In this work, they propose a benchmark named RepoBugs, which includes 124 bugs from open source repositories to evaluate the performance of LLMs.

Fine-tuning is also necessary to unlock state-of-the-art performance in program repair. MORepair (Yang et al. 2024c) is a multi-objective fine-tuning approach that instructs LLMs to generate high-quality patches. It involves adapting the LLM parameters to the syntactic nuances of code transformation and specifically fine-tuning the model to understand the logical reasoning behind code changes in the training data. This fine-tuning strategy enables LLM to achieve superior performance in program repair. de-Fitero-Dominguez et al. (2024) fine-tunes LLM on datasets containing C code vulnerabilities. They specifically design a structured representation of the code and provide it to LLM, including the line number of the code that needs to be repaired, the vulnerability description (i.e., CWE description), and the complete source code. The output of LLM is also structured and can be directly patched, which enables the code to be repaired automatically without manual intervention. Zhao et al. (2024d) explores how LLMs can achieve excellent APR performance through process supervision and feedback. They first construct a dataset called CodeNet4Repair, which is filled with multiple repair records for supervised fine-tuning. Then,they develop a reward model that provides feedback on the fine-tuned LLM's actions, progressively optimizing its policy for better repair. Dehghan et al. (2024) proposes continual merging and empirically studies the capabilities of merged adapters in Code LLMs for the APR task. Specifically, task-specific adapters are first trained for the LLM, and then MergeRepair is used to merge multiple task-specific adapters, considering the order and weight of the merged adapters for better APR.

Target-specific program repairing

We also investigate some studies on program repairing for some specific targets. Tol and Sunar (2023) proposes a framework called ZeroLeak, which explore how LLMs can be used to automatically generate repair code to address side-channel vulnerabilities in software. ZeroLeak guides LLMs to generate patches for specific vulnerabilities through zero-shot learning. Once generated, these patches are inspected by dynamic analysis tools to ensure that they not only function correctly, but also prevent information leakage. Paria et al. (2023) introduces a novel framework named DIVAS. The framework maps user-defined SoC specifications to Common Weakness Enumerations (CWEs), generates SystemVerilog Assertions (SVAs) for verification, and enforces security policies. DIVAS automates the process of vulnerability detection and policy enforcement, reducing manual effort and enhancing SoC security. Ahmad et al. (2023b) constructs a corpus of hardware security vulnerabilities and utilize LLMs to automatically remediate Verilog code containing these vulnerabilities. Charalambous et al. (2024) focuses on the software implementation of neural networks and related memory safety issues, including NULL pointer dereferencing, out-of-bounds access, double-free errors, and memory leaks. They propose detecting these vulnerabilities and automatically repairing them with the help of LLMs. Le et al. (2024) focuses on the application of LLMs (e.g., ChatGPT and Bard) in repairing security vulnerabilities in JavaScript programs. Using the top 25 CWEs of 2023 as a reference, they selecte JavaScript-related vulnerabilities to evaluate the accuracy of the models in generating the correct patches. Their findings highlight the potential of LLMs for JavaScript security, emphasizing the effectiveness of LLMs for programming languages used for web development. To convert a regular C/C++ program into its

Table 11 LLM assisted cyber attack

Category	Related work
Penetration testing (4)	Deng et al. (2023a), Happe and Cito (2023), Huang and Zhu (2023), Pratama et al. (2024)
Full-life-cycle cyberattack (3)	Xu et al. (2024b), Wang et al. (2024e), Usman et al. (2024)
Phishing website/email generation (3)	Begou et al. (2023), Roy et al. (2024), Francia et al. (2024)
Privilege escalation attacks (1)	Happe et al. (2023)
Payload generation (1)	Charan et al. (2023)
Attack graph generation (1)	Prapty et al. (2024)
CTF challenges (3)	Tann et al. (2023), Shao et al. (2024), Zhang et al. (2024a)
Proxies for attacks (1)	Beckerich et al. (2023)

HLS-compatible counterpart (HLS-C), Xu et al. (2024c) proposes an LLM-driven program repair framework that takes standard C/C++ code as input and automatically generates the corresponding HLS-C code for synthesis, minimizing human repair effort.

LLM assisted attack

A report (Barrett et al. 2023) from the workshop organized by Google on January 1, 2024 highlight the dual-use issue of Generative Artificial Intelligence (GenAI). These techniques can be used for both positive purposes and potentially for malicious attacks. In this section, we discuss current attacks with the help of LLMs in detail (Table 11).

Current status of LLM-assisted attacks

Sharma and Dash (2023) points out that ChatGPT has both positive and potentially negative impacts on cybersecurity. They list various types of threats to cybersecurity today, including malware attacks, phishing, and password attacks. They also mention the potential application of ChatGPT in social engineering attacks. Gupta et al. (2023) also conduct similar work on the impact of generative AI in cybersecurity and privacy. Furthermore, Moskal et al. (2023) explores the potential of LLMs for network threat testing, particularly in supporting threat-related actions and decisions. Experimenting on virtual machines, they discuss in detail how automated attacks guided by LLMs can be launched against devices in a network. They conclude that while this work is preliminary, it demonstrates that LLMs shows strong potential for cyber threats. For existing accessible malicious LLMs, Lin et al. (2024) conducts a systematic study of 212 real-world Malla (malicious LLM), revealing how they spread and work in the underground market. They examine in detail the Malla ecosystem, development frameworks, exploitation techniques, and the effectiveness of Malla in generating various malicious content. They also provide insights into how cybercriminals utilize LLMs and strategies for combating such cybercrime.

Specifically, there are various means of executing automatic attacks with the help of LLMs.

LLM-enabled automated penetration testing

Deng et al. (2023a) introduces a tool called PentestGPT designed to perform automated penetration tests. PentestGPT consists of three modules: inference, generation and parsing. Each module reflects a specific role in the penetration testing team so that the system can more realistically simulate automated penetration tests. Happe and Cito (2023) also conducts a study on penetration testing with the help of LLMs. The study investigates two use cases: high-level task planning for security testing and low-level vulnerability hunting within vulnerable virtual machines. They cerate a feedback loop between LLM-generated operations and the virtual machine, allowing LLMs to analyze the state of the system to find vulnerabilities and suggest attack vectors. Huang and Zhu (2023) points out the importance of integrating penetration testing with vulnerability remediation into a cohesive system. They proposes PenHeal, a two-stage LLM-based framework designed to autonomously identify and mitigate security vulnerabilities. The framework integrates two LLM-enabled components: the Pentest Module, which detects multiple vulnerabilities within a system, and the Remediation Module, which recommends optimal remediation strategies. Pratama et al. (2024) developes CIPHER (Cybersecurity Intelligent Penetration-testing Helper for Ethical Researchers), a LLM trained using over 300 high-quality write-ups of vulnerable machines, hacking techniques, and documentation of open-source penetration testing tools. Additionally, they introduce the Findings, Action, Reasoning, and Results (FARR) Flow augmentation to enhance penetration testing write-ups, establishing a fully automated pentesting simulation benchmark tailored for LLMs.

LLM-assisted automatic full-life-cycle cyberattack

Xu et al. (2024b) proposes AUTOATTACKER, a system that leverages LLMs to automate the execution of "keystroke-operated" cyberattacks that mimic human operations. The system employs LLMs to generate precise attack commands for various techniques and environments, transforming potential manual operations into automated and efficient processes. AUTOATTACKER consists of multiple modules that interact iteratively with the LLM to construct complex attack sequences using functions such as summarization, planning, and action selection. AURORA (Wang et al. 2024e) is another automatic end-to-end framework for cyberattack construction and emulation. It can autonomously build multi-stage cyberattack plans based on CTI reports, construct the emulation infrastructure, and execute the attack procedures. Usman et al. (2024) introduces Occupy AI, a customized and fine-tuned LLM specifically designed to automate and execute cyberattacks. This specialized AI-driven tool is proficient in crafting attack steps and generating executable code for various cyber threats, including phishing, malware injection, and system exploitation.

LLM-assisted phishing website/email generation

Begou et al. (2023) uses LLM to automatically generate advanced phishing attacks. In the proposed attack method, LLMs are used for the following functions:

cloning target websites, modifying login forms to capture credentials, obfuscating code, automating domain name registration, and automating script deployment. Roy et al. (2024) examines the potential of LLMs like ChatGPT, GPT-4, Claude, and Bard to generate phishing attacks. The study finds that these models can effectively create convincing phishing websites and emails, mimicking well-known brands and employing evasive tactics to avoid detection. The research also develops a BERT-based detection tool that achieves high accuracy in identifying malicious prompts, serving as a countermeasure against the misuse of LLMs for phishing scams. Francia et al. (2024) compares the effectiveness of smishing (SMS phishing) messages created by GPT-4 and human authors, demonstrating that LLM-generated messages are generally perceived as more convincing than those authored by humans. The study also finds that targets are unable to identify whether a message was AI-generated or human-authored and struggle to pinpoint criteria that could help make this distinction. This poses a challenge against personalized AI-enabled social engineering attacks.

LLM-assisted privilege escalation attacks

Happe et al. (2023) uses LLM to assist in completing penetration tests. They develop an automated Linux privilege escalation benchmark to evaluate the performance of different LLMs. At the same time, they design a tool called Wintermute to quickly explore the ability of LLMs to bootstrap privilege escalation.

LLM-assisted payload generation

Charan et al. (2023) proposes to write payloads with the help of LLMs to launch cyber attacks. This study shows the high efficiency of LLMs by generating executable code for the top 10 MITRE weaknesses observed in 2022 using ChatGPT and Bard respectively. In addition, LLM-generated payloads tend to be more complex and targeted than manually crafted payloads.

LLM-assisted attack graph generation

Prapty et al. (2024) explores the approach of leveraging LLMs to automate the generation of attack graphs by intelligently chaining CVEs based on their preconditions and effects. They also show how to utilize LLMs to create attack graphs from threat reports.

LLM-assisted capture the flag (CTF) challenges

Tann et al. (2023) investigates the potential of existing LLMs in solving CTF competitions. They select a number of representative challenges from common CTF categories to evaluate the performance of LLMs, including GPT—3.5, PaLM2, and Prometheus. Their research results demonstrate that LLMs can indeed help participants cope with CTF challenges to a certain extent, albeit not comprehensively.

Proxies for attacks Beckerich et al. (2023) uses ChatGPT as a proxy between the victim and the network controlled by the attackers (C&C), which allows the attacker to remotely control the victim's system without communicating directly, making it difficult to track down the attackers.

Table 12 LLMs for (In)secure code generation

Category	Related work
Generate Code Evaluation (11)	Sandoval et al. (2023), Tambon et al. (2024), Tihanyi et al. (2024a), Pearce et al. (2022a), Wang et al. (2024d), Wang et al. (2023f), Liu et al. (2024g), Siddiq and Santos (2023), Liu et al. (2024a), Ullah et al. (2023), Buscemi (2023)
Secure Code Generation (7)	Khoury et al. (2023), Kavian et al. (2024), He and Vechev (2023), Li et al. (2024b), He et al. (2024), Tang et al. (2024), Wong et al. (2024)

(In)secure code generation

There have been many previous works that have confirmed that LLMs do have good code comprehension capabilities (He et al. 2024; Luo et al. 2024; Roziere et al. 2023; Li et al. 2023d). However, the security of the generated code is very important, and some studies have explored this issue (Table 12).

Evaluation of the security of LLM-generated code. It is very important to know whether the code generated by LLMs has security risks. Sandoval et al. (2023) conducts an experiment to explore whether code written by undergraduate computer science students with the help of LLMs poses any additional security risks. Participants are tasked with implementing a singly-linked 'shopping list' structure in C and they are divided into two groups: a control group that doesn't have access to Codex, and an assisted group that does. The results show that LLM does not significantly increase the risk of introducing security vulnerabilities when used as a code assistant. Tambon et al. (2024) conducts an empirical study investigating bugs in code generated by LLMs, focusing on three models: CodeGen, PanGu-Coder, and Codex. The research identifies 10 unique bug patterns among 333 collected errors, and these patterns are confirmed by 34 LLM practitioners and researchers. Tihanyi et al. (2024a) study how LLMs generate vulnerabilities when writing simple C programs using a neutral zero-shot prompt. They collected code generated by Gemini-pro, GPT-4, Falcon-180B, CodeLLama2-13B, and other LLMs under neutral prompts, which constitute the FormAI-v2 dataset. The study found that at least 63.47% of the generated programs are vulnerable, highlighting the risks of using LLM-generated code.

There are many studies exploring the security of code generated by state-of-the-art LLMs. Pearce et al. (2022a) investigates the security of code generated by GitHub Copilot. They design 89 different execution scenarios for Copilot, resulting in 1,689 programs. These programs are then analyzed for vulnerabilities, particularly focusing on the top 25 CWEs identified by MITRE. Wang et al. (2024d) introduces CodeSecEval, a meticulously curated dataset designed to address 44 critical vulnerability types with 180 distinct samples. The dataset is then used for precisely evaluating and enhancing the security aspects of code generated by LLMs. The study reveals that current models frequently overlook security issues during both code generation and repair processes, leading to the creation of vulnerable code. Wang et al. (2023f) delves into the potential of LLMs in security-oriented program analysis. Their evaluation focuses on two representative LLMs, ChatGPT and CodeBERT, evaluating their performance on analysis tasks of varying difficulty, including vulnerability analysis, bug fixing, fuzzing, and assembly code analysis. Liu et al. (2024g) evaluates the code generated by ChatGPT, focusing on aspects such as correctness, understandability, and security. Through an empirical study using LeetCode questions and CWE scenarios, they analyze the quality of code snippets generated by ChatGPT and its ability to improve the code through multi-round dialogue. The results reveal that while ChatGPT is able to generate functionally correct code, it encounters challenges in complex reasoning and ensuring code security.

On the other hand, Siddiq and Santos (2023) proposes a framework called SALLM specifically for evaluating the security of code generated by LLMs. SALLM consists of three components: a prompt dataset detailing Python programs, a code generation environment that requires different solutions from LLMs, and a systematic evaluation model that leverages Docker to execute the generated code. Liu et al. (2024a) focuses on enhancing the quality evaluation of code generation. Recognizing that existing benchmarks often have a limited set of test cases, they introduced a code synthesis evaluation framework, EvalPlus. EvalPlus significantly expands the number of test cases in the evaluation dataset by deploying an automatic test input generator that combines LLMs with a mutation-based strategy. Ullah et al. (2023) collects 228 code scenarios and analyze 8 LLMs in an automated framework to determine whether LLMs can reliably identify security-related vulnerabilities. They point out that current LLMs fall short in automated vulnerability detection tasks and outline several limitations exhibited by current LLMs. Buscemi (2023) evaluates the performance of ChatGPT—3.5 on generating code, including an examination of code security in 10 programming languages.

Do LLMs know whether the generated code is safe or not?Khoury et al. (2023) conducts a series of experiments to evaluate the security of LLM-generated code and to discover vulnerabilities in generated code under various scenarios. The results show that while LLMs may identify vulnerabilities in the generated code when prompted for review, they still generate unsafe code unless explicitly instructed otherwise. A significant challenge they faced stems from the uninterpretability of deep neural networks, which causes LLMs to give inconsistent responses when repeatedly asked about code security, without a clear strategy to maximize successful identification.

To ensure the generation of secure codes, LLM-SecGuard (Kavian et al. 2024) enhance code security through the synergy between static code analyzers and LLMs. He and Vechev (2023) takes a more direct approach to customize LLMs through specific mechanisms. They propose a method named svGen, which makes LLMs generate safe or unsafe code based on the user's security preferences. In addition to the descriptions for the generated code, they also introduce property-specific continuous vectors (called prefixes), which are sequences of vectors that match the shape of the LLMs' hidden states. These prefixes are optimized to influence the LLM's generation process by setting initial hidden states that steer the code toward meeting the desired security criteria, all without modifying the underlying weights of the LLM.

Fine-tuning LLMs for secure code generation is feasible. Li et al. (2024b) reveals that fine-tuning LLMs can improve secure code generation by 6.4% for C language and 5.4% for C++ language. Additionally, fine-tuning with function-level and block-level datasets achieves the best performance in secure code generation, compared to file-level and line-level datasets. He et al. (2024) introduces SafeCoder, an innovative instruction tuning approach that enhances the security of code generation by LLMs. SafeCoder combines traditional instruction tuning with security-specific fine-tuning using a high-quality dataset collected through an automated pipeline from GitHub. This approach significantly enhances code security without compromising the LLMs' utility across various tasks, demonstrating its adaptability and effectiveness in enhancing the security of LLM-generated code.

In addressing the question of how to best iteratively refine code, Tang et al. (2024) points out that the process exposes an explore-exploit tradeoff, which can be framed as a multi-armed bandit problem, and solved using Thompson Sampling. The resulting LLM-based program synthesis algorithm is widely applicable. Wong et al. (2024) discusses iterative code repair in both high and low-resource languages, where an LLM fixes an incorrect program by reasoning about errors and generating new code. Specifically, they delve into guiding the model to generate secure code through chain-of-thought reasoning.

Others

Apart from the previously described categories, there are a few scattered studies on the application of LLMs in the field of cybersecurity, which are also of research value.

IoT fingerprint

Sarabi et al. (2023) proposes a method for Internet devices fingerprint generation. Their approach is divided into two steps. First, raw text data obtained from web scans is converted into a stable embedded representation with RoBERTa. Next, the embedding is clustered using the HDBSCAN and the fingerprint is generated based on the clustering.

Botnet

Yang and Menczer (2023) introduces a LLM-driven botnet called fox8 on Twitter. The fox8 botnet contains over one thousand users controlled by AI. They post machine-generated content and stolen images to spread fake and harmful information, engaging with each other through replies and retweets.

Security patch detection

Tang et al. (2023) proposes a system named LLMDA, whose main goal is to improve the identification of security patches in open-source software (OSS). LLMs are used to generate explanatory descriptions of patches and synthetic data, which helps to augment existing datasets.

SoC security

Saha et al. (2024) explores the potential of integrating LLMs into the system-on-chip (SoC) security verification paradigm. They provide a systematic evaluation of LLM applications about vulnerability insertion, security assessment, security verification, and countermeasure development.

Taint analysis

Liu et al. (2023d) introduces LATTE, a static binary taint analysis tool supported by LLMs. LLMs help to identify the chain of data dependencies between taint sources and possible vulnerability triggers. LLMs could provide an understanding of code structure and semantics in the process.

LLMs' input–output safeguard

Inan et al. (2023) proposes Llama Guard to detect the risk in LLM's prompt and response. Using labeled security risk text, they perform instruction tuning on Llama2-7b to obtain this model.

Honeypot

Sladic et al. (2024) designs a dynamic and real-time fake honeypot by giving response generated by LLMs, which mainly focus on changing the limitation that honeypots are easily recognizable. In their experiment, most people can't recognize whether the remote host is a real one or a honeypot generated by LLMs. Reti et al. (2024) systematically investigates the use of LLMs to create a variety of honeytokens. They design different types of honeytokens to evaluate the optimal prompts, including configuration files, databases, and log files. They test 210 different prompt structures, based on 16 prompt-building blocks, and demonstrate that LLMs can generate a wide array of honeytokens using the presented prompt structures. LLMPot (Vasilatos et al. 2024) is a novel approach for designing honeypots in ICS networks that harnesses the power of LLMs. It aims to automate and optimize the creation of realistic honeypots with vendor-agnostic configurations, applicable to any control logic, thereby eliminating the manual effort and specialized knowledge traditionally required.

Incidence response

Hays and White (2024) advocates for the application of ChatGPT to enhance incident response planning (IRP) in cybersecurity. It suggests that LLMs can draft initial plans, recommend best practices, and identify documentation gaps. The paper highlights the potential of LLMs to streamline IRP processes, emphasizing the value of human oversight to ensure accuracy and relevance.

Network management

Mani et al. (2023) explores how LLMs can be used to generate task-specific code from natural language queries to improve network management. They develop and release a test benchmark, NeMoEval, covering two network management applications: network traffic analysis and network lifecycle management.

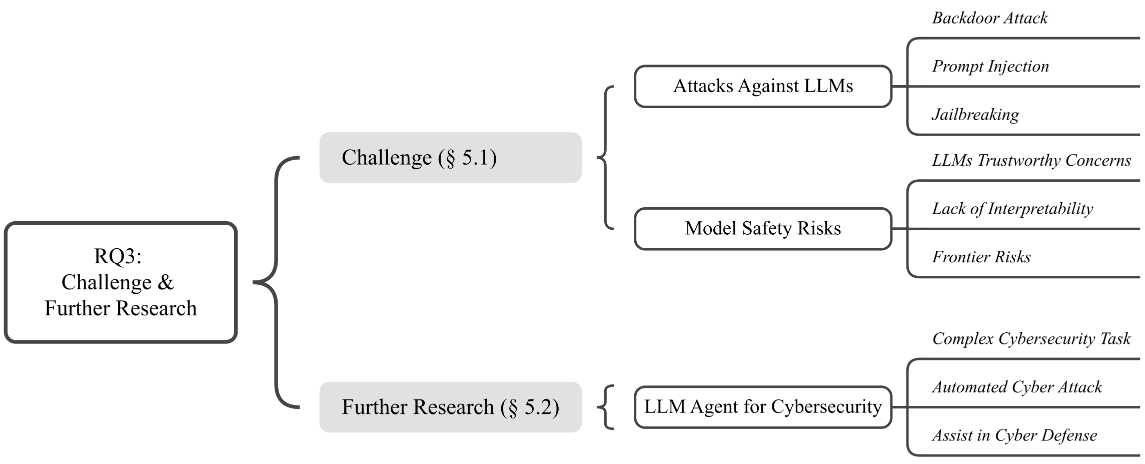


Fig. 5 An overview of RQ3

Table 13 LLM agents for cybersecurity

Category	Related work
Complex Tasks (3)	Cui et al. (2024), Rigaki et al. (2024), Huang et al. (2023b)
Cyber Attacks (4)	Fang et al. (2024c), Moskal et al. (2023), Fang et al. (2024b), Fang et al. (2024d)
Cyber Defense (4)	An et al. (2024), Kaheh et al. (2023), Cao et al. (2024), Tseng et al. (2024b)

Vulnerabilities reproduction
Feng and Chen (2024) proposes an approach called AdbGPT that utilizes LLMs to automatically reproduce vulnerabilities in vulnerability reports by prompting engineering without training or hard coding.

Expertise Q&A on cybersecurity domain
Kabir et al. (2024) conducts an empirical study of ChatGPT's performance in answering Stack Overflow programming questions. The main drawbacks of the LLM answers are fake information and excessive length of the content. Still some testers like its comprehensiveness and good style of language presentation. Due to the difficulty of recognizing misleading information given by LLMs, this is an area that has yet to be researched.

Answer to Q2: LLMs have shown great potential in the field of cybersecurity, assisting in various aspects such as threat intelligence, anomaly detection, vulnerability detection, and so on. LLM security copilot can effectively empower the automation and intelligence of cybersecurity, helping to address security risk challenges. Although relevant research has made certain progress, it is still worth further exploration to better apply LLMs in the field of cybersecurity.

RQ3: What are the challenge and further research for the application of LLMs in cybersecurity?

Challenge

The application of LLMs in cybersecurity represents a cutting-edge field, demonstrating the power of LLMs in dealing with complex and dynamic cyber threats. However, despite their strengths, LLMs are not without challenges, especially their inherent vulnerabilities and susceptibilities to attacks (Yao et al. 2024b; Zhao et al. 2024c). Among the critical concerns are the phenomena of LLMs-oriented attacks and LLMs jailbreaking. These vulnerabilities highlight the double-edged nature of LLM applications in cybersecurity (Pasupuleti et al. 2023). On one hand, the powerful comprehension and predictive capabilities of LLMs can significantly promote the intelligence of cybersecurity systems. On the other hand, their intrinsic weaknesses facilitate exploitation and pose serious security risks, undermining their reliability and integrity in cybersecurity applications.

In this part, we discuss attacks against LLMs and model risks separately, depending on whether the security challenges arise from intentional attackers or from risks inherent in the model itself (Fig. 5).

Attacks against LLMs

The vulnerabilities of LLMs make them susceptible to attacks by malicious users (Kumar et al. 2024; Esmradi et al. 2023; Wu et al. 2024b). There are various types of attack against LLMs, including backdoor attack, prompt injection, and jailbreaking.

Backdoor attack manipulates model outputs to achieve attackers' objectives by embedding specific triggers in the model or its inputs. Shi et al. (2023) proposes a novel backdoor attack methodology called BadGPT, specifically targeting language models that have been fine-tuned through reinforcement learning, such as ChatGPT. This approach involves embedding backdoors within the reward model, which can be activated via specific trigger prompts. Such activation allows attackers to control the model's output to align with their preferences, showcasing a critical security vulnerability. In another study, Zhao et al. (2024b) introduces a novel backdoor attack strategy, ICLAttack, which aims at exploiting the inherent context learning capabilities of LLMs. The ICLAttack framework encompasses two primary attack vectors: poisoning demonstration examples and poisoning demonstration prompts. By embedding backdoor triggers within the model's context, ICLAttack is able to influence the model's behavior without the need for fine-tuning, thus revealing universal vulnerabilities within LLMs. Furthermore, Yao et al. (2024a) reveals a backdoor attack mechanism tailored to prompt-based LLMs, called PoisonPrompt. The method injects backdoors into the language model through two steps: poisoned prompt generation and bi-level optimization. PoisonPrompt can alter the normal prediction of the model in case of specific trigger activations without affecting the performance of the model on downstream tasks, posing a subtle but powerful threat to the integrity of LLMs.

Prompt injections involve attackers inserting special commands into inputs, compelling the model to execute actions aligned with the attackers' intentions. Pedro et al. (2023) conducts a comprehensive investigation of prompt-to-SQL (P2SQL) injection attacks against web applications based on the Langchain framework. These attacks utilize user-input prompts to generate malicious SQL queries, thereby enabling attackers to tamper with databases or steal sensitive information. Jiang et al. (2023b) introduces the Compositional Instruction Attack (CIA), unveiling the susceptibility of LLMs to attacks that utilize synthetic instructions with potentially malicious intentions. Through two transformation methods, Talking-CIA and Writing-CIA, harmful instructions are masked as conversational or writing tasks, preventing the model from recognizing potentially malicious intent and thus generating harmful content. Liu et al. (2023f) proposes a novel black-box prompt injection attack technique named HOUYI for applications integrated with LLMs. HOUYI executes attacks through three key elements: pre-constructed prompts, injection prompts, and malicious payloads. Its deployment across 36 real-world scenarios demonstrates its efficacy in discovering and exploiting vulnerabilities within LLM-integrated applications. Yan et al. (2024a) focuses on Virtual Prompt Injection (VPI) attacks against instruction-tuned LLMs, which allow attackers to manipulate model behavior by specifying virtual prompts without directly injecting into model inputs, leading to the model disseminating biased information. Piet et al. (2024) uses instruction-tuned models to generate datasets for specific tasks. These datasets are then utilized to fine-tune foundational models, enhancing their robustness to resist most prompt injection attacks. Additionally, Kour et al. (2023) constructs an adversarial attack dataset named AttaQ in a semi-automated manner, aiming to evaluate the security of LLMs in the face of harmful or inappropriate inputs. Vulnerabilities are exposed by analyzing model responses to the AttaQ dataset, and specialized clustering techniques are further applied to identify and characterize the models' vulnerable semantic areas.

Jailbreaking refers to the phenomenon of LLMs generating unsafe or unintended content when prompted in certain ways, despite being designed with safeguards (Chu et al. 2024; Xu et al. 2024e). Owing to the advancing capabilities of LLMs, this issue has attracted significant attention in recent years. Shen et al. (2023) studies the security issues of LLMs when facing jailbreak prompts. They collect and analyze 6,387 prompts to reveal the characteristics and attack strategies of these prompts. Despite various security measures implemented by LLMs, they found that effective jailbreak prompts still successfully induce models to generate harmful content, indicating the need for further improvements in the security of LLMs. Chu et al. (2024) conducts a comprehensive evaluation of LLMs jailbreaking, revealing the effectiveness of these attack methods and the vulnerabilities of LLMs across various violation categories.

There are various methods for generating adversarial prompts for jailbreaking. Zou et al. (2023) combines greedy search and gradient-based optimization techniques to propose a method that automatically generates adversarial suffixes to prompt models, both open-source and commercial, to produce inappropriate content. Lapid et al. (2023) introduces a novel approach to black-box jailbreak attacks using genetic algorithms, which can manipulate LLMs to produce unexpected and potentially harmful outputs without accessing the model's internal structure and parameters by optimizing a universal adversarial prompt. Ding et al. (2024) conceptualizes the jailbreaking process as prompt rewriting and scenario nesting. They then introduce ReNeLLM, a jailbreaking prompt generation framework that utilizes LLMs to generate effective jailbreaking prompts. Compared to existing baselines, ReNeLLM achieves high attack success rates on multiple LLMs while significantly reducing the time cost. Deng et al. (2024) explores jailbreak attacks on LLM Chatbots and proposes a framework named MASTERKEY to automate this process. Through temporal feature analysis and automated prompt generation, MASTERKEY reveals and bypasses the defense mechanisms of LLM chatbots, offering new perspectives for LLM security research and guidance for service providers to improve their security measures.

Research on LLMs jailbreaking can also be used for red-teaming. Zhu et al. (2024) proposes AutoDAN, an interpretable and gradient-based adversarial attack method. By combining the dual objectives of jailbreaking and readability, it generates interpretable and diverse attack prompts capable of effectively bypass perplexity filters and demonstrates robust generalization in scenarios with limited training data. This method not only offers a novel approach for red-teaming of LLMs but also helps to understand the mechanics of jailbreak attacks. Yu et al. (2023) presents a new black-box jailbreak fuzzing framework named GPTFUZZER. By collecting human-written jailbreak templates from the internet as initial seeds, and then iterating through a process of seed selection, mutation, and evaluating the success of attacks, GPTFUZZER significantly enhances the efficiency and scalability of red team testing. Yao et al. (2023a) introduces FuzzLLM, a novel and universally applicable fuzz testing framework designed to proactively discover jailbreak vulnerabilities in LLMs. FuzzLLM employs a template-based strategy that generates a variety of jailbreak prompts and identify potential security vulnerabilities through automated testing. It demonstrates efficiency and comprehensiveness across various LLMs, effectively identifying and assessing jailbreak vulnerabilities.

Additionally, Wang et al. (2023e) introduces the concept of a semantic firewall to describe the defense mechanisms of LLMs against malicious prompts and proposes a self-deception attack method to bypass LLMs semantic firewalls. This method designs a customizable dialogue template for experimenting with specific illegal payloads and automatically achieving LLM jailbreak. Qiu et al. (2023) develops a potential jailbreak prompt dataset embedded with malicious instructions and proposes a hierarchical annotation framework to analyze the performance of LLMs under different conditions(e.g., instruction positions, word substitutions, and instruction replacements). This is aimed at evaluating the security and output robustness of LLMs when processing texts containing potential malicious instructions. Li et al. (2023b) investigates the potential privacy threats associated with ChatGPT and the Bing search engine integrated with ChatGPT. By introducing a novel multi-step jailbreaking prompt, they successfully extract personally identifiable information from ChatGPT and demonstrate the privacy threats posed by the new Bing under direct prompts.

Model safety risks

Even in the absence of direct adversarial attacks, inherent risks within these models limit their application in cybersecurity, including LLMs trustworthy concerns, lack of interpretability, and frontier risks.

LLMs trustworthy Concerns The surge of LLMs brings significant concerns regarding their trustworthiness, especially considering the inherent risks in the models themselves, which pertain to the aspects and extent to which humans can trust AI. Existing research in AI governance and trustworthy LLMs has provided guidance for the concern dimensions of trustworthy LLMs (Tabassi 2023; Liu et al. 2023a; Wang et al. 2023a).

Hallucination is a response generated by AI that contains false or misleading information presented as fact (Maynez et al. 2020; Ji et al. 2023). Ensuring the authenticity of content generated by language models is a critical issue that requires urgent attention. In practical applications, the content produced by LLMs may exhibit factual hallucinations, which severely impact the reliability of their outputs. Several studies have explored the causes of hallucinations and proposed mitigation strategies (Huang et al. 2023a; Zhang et al. 2023e). The causes of hallucinations are typically attributed to issues arising during the data, training, and inference stages, such as poor data quality, misinformation, outdated knowledge, flaws in model architecture and strategies, and randomness in the inference process. Although hallucinations are difficult to eliminate completely (Xu et al. 2024d), they can be mitigated through various methods, such as building high-quality datasets, optimizing decoding strategies, and enhancing external knowledge through techniques like Retrieval-Augmented Generation (RAG). Addressing the hallucination issue is of significant importance for improving the trustworthiness of LLMs in cybersecurity applications.

Toxicity in language models is characterized as rude, disrespectful, or unreasonable commentary that is likely to drive individuals away from a discussion (Welbl et al. 2021). This is an inherent property of LLMs, stemming from their inevitable exposure to toxic content during training. For instance, an analysis of the LLaMA2 pretraining corpus revealed that approximately 0.2% of the documents could be identified as toxic content (Touvron et al. 2023b). The detoxification of LLMs can be broadly categorized into two approaches. The first category involves internal modifications to the model to prevent the generation of toxic content, such as employing Reinforcement Learning with Human Feedback (RLHF) to align the model with safety guidelines (Ouyang et al. 2022) or employing knowledge editing techniques to precisely modify toxic regions within the LLMs (Wang et al. 2024f). The second approach involves using external classifiers to filter the model's outputs (Inan et al. 2023). In summary, reducing the toxicity of generated content is essential to preventing harm to individuals, groups, and broader societies.

Fairness in LLMs encapsulates the ethical principle that necessitates the equitable design, training, and deployment of LLMs and related AI systems, preventing biased or discriminatory outcomes (Wang et al. 2023c). Language models may exhibit discrimination and bias, primarily due to the characteristics of their training data and model design (Li et al. 2023e). The training data collected from the internet reflects real-world biases, including those related to race, gender, culture, religion, and social status. It is difficult to completely filter and clean all biased content. Additionally, in the design of generative AI models, there is a lack of effective mechanisms to mitigate biases, which results in the models capturing discriminatory patterns from the training data. In response, various strategies have been proposed to improve fairness in LLMs, ranging from holistic approaches to mitigating specific types of biases, such as biases in internal components of LLMs and biases arising from user interactions (Dev et al. 2023; Dong et al. 2024), thereby promoting AI models' adherence to fairness and anti-discrimination principles.

Privacy means the norms and practices that help to safeguard human and data autonomy, identity, and dignity (Tabassi 2023). Unlike previous concerns that primarily focused on sensitive data protection, advanced generative AI models trained on massive datasets posses sophisticated memory mechanisms that may result in privacy leaks (Carlini et al. 2021, 2022; Peris et al. 2023). And various privacy protection methods have been developed for AI models, including data anonymization, federated learning, differential privacy, and unlearning, each addressing distinct privacy challenges (Murthy et al. 2019; Nagy et al. 2023; Vasa and Thakkar 2023; Sekhari et al. 2021). Nevertheless, privacy protection for generative AI is still in its early stages, and exploring how to fully utilize the capabilities of these models while safeguarding personal privacy remains a challenge.

Robustness in LLMs refers to their stability and performance when faced with various input conditions. This includes their ability to effectively handle diverse inputs, noise, interference, adversarial attacks, and changes in data distribution, among other factors (Huang et al. 2024). In addition to the model's performance against malicious attacks discussed in the previous chapter, the performance of the model on noisy data and out-of-distribution (OOD) data is also used to assess robustness (Wang et al. 2021, 2024c). This capability is crucial for LLMs in real-world applications, as it enables the models to respond appropriately when dealing with unknown or new inputs. Therefore, the application of LLMs in the cybersecurity domain requires a high level of robustness.

Beyond the key dimensions discussed above, there are other important aspects of model trustworthiness that warrant attention, including accountability, machine ethics, environmental well-being, data governance, reproducibility, and human oversight, among others. These are crucial for building trustworthy and responsible AI systems, and significantly impact the application of models in specific areas such as cybersecurity.

Lack of Interpretability Interpretability refers to the ability of an AI system to explain its decisions and outputs in a manner understandable to humans. The underlying principle is to design models or algorithms that can generate corresponding explanations when making predictions or decisions (Zhang et al. 2021).

As AI systems grow increasingly advanced, it has become challenging for humans to understand and trace how algorithms produce their results. The computational process has thus evolved into what is commonly referred to as a "black box" - a system with opaque internal mechanisms. The complexity of these systems and their lack of transparency pose significant risks, especially in sensitive and critical domains such as cybersecurity.

Research into the interpretability of AI models not only guides model improvement and optimization but also enhances user trust in their safe application. Studies explore interpretability from various perspectives, including the data level (analyzing inputs, outputs, datasets, and data modalities), the model level (examining tokens, features, neurons, network layers, and architecture), and the training and reasoning process (investigating how models are trained and how they perform during inference) (Dang et al. 2024). For instance, OpenAI uses GPT-4 to generate natural language explanations of neuron behaviors in GPT-2 and rates these explanations (Bills et al. 2023). Anthropic employs dictionary learning to isolate repeated neuron activation patterns across different contexts, aiding in understanding how concepts are represented in language models (Templeton et al. 2024). Additionally, studies on circuits and sparse autoencoders have been conducted to uncover the behaviors of black-box models (Elhage et al. 2022, 2021; Huben et al. 2023; Gao et al. 2024).

Despite numerous attempts to explain neuron behaviors, little progress has been made in understanding the underlying mechanisms that generate these behaviors (Bills et al. 2023). The black-box nature of AI models remains largely unresolved. Moreover, as the number of model parameters increases and more complex emergent behaviors arise, the lack of interpretability will persist as a significant challenge in AI development and application.

Frontier risks

With the advancement of AI, some researchers have raised concerns about the potential catastrophic risks posed by AI. Carlsmith (2022) highlights that imperfectly controlled agents may deliberately seek power over humans, and such power-seeking AIs could result in human disempowerment, leading to catastrophic outcomes. Similarly, issues such as proxy gaming (Clark and Amodei 2016) and goal drift (Hendrycks et al. 2023) could cause highly intelligent AIs to lose control, further exacerbating these risks. Hubinger et al. (2024) investigates the deceptive behaviors that LLMs may exhibit under specific trigger conditions, finding that these behaviors could persist even after safety alignment, thereby posing a potential threat to AI system security.

Regarding emerging frontier risks, Li et al. (2024c) and Stewart (2024) discuss how LLMs could contribute to the proliferation of chemical, biological, radiological, and nuclear weapons. Given that LLMs have been trained on vast amounts of computer code and possess the ability to generate scripts and code, they could facilitate engineering design and computer simulations related to specific CBRN production processes. Therefore, caution is necessary when deploying LLMs in sensitive fields, and it is critical to implement mitigation measures to prevent the generation of problematic outputs.

Answer 1 to Q3: Despite the powerful capabilities of LLMs, they inherently possess certain weaknesses and vulnerabilities, posing significant challenges to the application of LLMs in cybersecurity.

Further research

Despite the significant research into LLMs within the field of cybersecurity, the exploration and application of such models remain in their initial stages and have great potential for development (de Jesus Coelho da Silva and Westphall 2024; Motlagh et al. 2024). The complexity of cybersecurity stems not only from the diversity of attack methods but also from the intricate nature of network environments, which requires the integrated application of various tools and strategies to achieve effective protection (Azizi and Haass 2023; Mtsweni et al. 2018). Facing these challenges requires AI systems to have stronger capabilities in planning, reasoning, tool use, and memory. Consequently, the concept of LLM Agent has emerged and attracted a lot of attention from researchers (Table 13).

LLM Agent is "a system that can use an LLM to reason through a problem, create a plan to solve the problem, and execute the plan with the help of a set of tools (Varshney 2023)." By simulating complex network behaviors and attack patterns, and integrating advanced natural language processing capabilities, LLM agents introduce new perspectives and solutions to the field of cybersecurity (Kaheh et al. 2023; Moskal et al. 2023; Cui et al. 2024; Rigaki et al. 2024; Fang et al. 2024c; An et al. 2024). With the continuous advancement of technology and in-depth research, LLM agents are expected to play a key role in defense strategy generation, threat detection, and security policy formulation, significantly improving the efficiency and intelligence level of cybersecurity defenses.

The AI Agents framework based on LLMs possesses the critical capabilities required to solve complex problems (Ruan et al. 2023). Xi et al. (2023) proposes an LLM Agent architecture that includes brain, perception, and action components to provide a wide range of applications in single-agent scenarios, multi-agent environments, and human-agent collaboration. Moreover, the incorporation of Tool & API calls endows LLM agents with the capacity to interact with the real world. Qin et al. (2024) develops the ToolBench dataset and the DFSDT algorithm to enable LLMs to successfully handle complex tasks involving numerous real-world APIs. Liu et al. (2024d) introduces a sophisticated tool invocation mechanism that enhances LLMs' interaction with external tools by summarizing and making decisions. Additionally, Yang et al. (2024d) demonstrates that integrating code into LLMs significantly enhances its ability to perform more complex tasks as an intelligent agent. Qiao et al. (2023) proposes TaskWeaver, a code-first agent framework for seamlessly planning and executing data analytics tasks.

LLM agents can be applied to address complex cybersecurity tasks. Cui et al. (2024) proposes an innovative framework named LLMind, which utilizes LLM as a coordinator to perform complex tasks by integrating with IoT devices and domain-specific AI modules. The framework employs finite state machine methods to generate control scripts, thereby enhancing the accuracy and success rate of task execution. In addition, LLMind introduces a mechanism for accumulating experience, which allows the system to continually learn and progress through ongoing interactions between users and machines. Rigaki et al. (2024) demonstrates the use of LLMs as agents within cybersecurity environments. Experiments show that LLM agents can achieve performance comparable to or better than extensively trained agents in sequential decision-making tasks, even without additional training. Furthermore, the study introduces the NetSecGame environment, a highly modular and adaptive cybersecurity environment designed to support complex multi-agent scenarios. Huang et al. (2023b) proposes ChatNet, a domain-specific network LLM framework with access to a variety of external network tools. ChatNet significantly reduces the time required for tedious network planning tasks, thereby greatly increasing efficiency.

LLM agents can be employed to perform automated attacks. Fang et al. (2024c) reveals the potential of LLM agents in cybersecurity attacks, particularly the capability of GPT-4 to autonomously conduct complex hacker attacks on websites without prior knowledge of vulnerabilities. The study shows that LLM agents have a success rate of up to 73.3% in hacking attempts and can autonomously discover vulnerabilities in real-world websites. Moskal et al. (2023) demonstrates the potential application of LLMs in cyber threat testing, especially in automating cyber attack activities. With prompt engineering and automated agents, LLMs can understand and execute complex cyber attacks. Fang et al. (2024b) collects a dataset of 15 zero-day vulnerabilities. Based on this dataset, the study shows that LLM agents can autonomously exploit these zero-day vulnerabilities in real-world systems. Fang et al. (2024d) also shows that teams of LLM agents can exploit real-world, zero-day vulnerabilities by designing a system of agents with a planning agent that can launch subagents.

LLM agents can also be utilized to assist in cyber defense. An et al. (2024) designs a multi-agent system (Nissist) to precisely understand user queries and provide effective mitigation plans. Nissist utilizes troubleshooting guides and incident mitigation history to provide suggestions, which significantly reduces the time for event mitigation, reduces the workload of on-duty engineers, and enhances the reliability of services. Cyber Sentinel (Kaheh et al. 2023) is a dialogue agent based on GPT-4, which can interpret potential cyber threats and execute security actions based on user instructions. The potential impact of Cyber Sentinel in cyber security includes improved threat detection and response capabilities, enhanced operational efficiency, real-time collaboration, and knowledge sharing. PhishAgent (Cao et al. 2024) is a multimodal agent that combines a wide range of tools, integrating both online and offline knowledge bases with Multimodal LLMs, showing strong resilience against various types of adversarial attacks. Tseng et al. (2024b) develops an AI agent to replace the labor intensive repetitive tasks involved in analyzing CTI reports. By leveraging the advanced capabilities of LLMs, the AI agent can accurately extract important information from large volumes of text and generate Regex to help SOC analysts accelerate the process of establishing correlation rules.

LLM agents enhance cybersecurity applications with their remarkable capabilities, yet the security risks inherent in agent systems (Yuan et al. 2024) pose challenges for their deployment in cybersecurity environments. Wu et al. (2024a) introduces the concept of Web-based Indirect Prompt Injection (WIPI), a novel cyber threat that embeds malicious instructions in web pages to indirectly control these agents, achieving high success rates and robustness across different user inputs. Zhan et al. (2024) highlights that LLM agents integration with external tools may lead to the risk of indirect prompt injection attacks, in which attackers embed malicious commands in the content processed by LLMs to manipulate these agents to perform actions harmful to users.

In conclusion, the application of LLM-based agents in cybersecurity opens up new avenues for dealing with cyber security threats. Although research in this area is still in its early stages, and the inherent security vulnerabilities of agents have not yet been addressed, this line of research promises to significantly enhance the capability to counter complex cyber threats and has the potential to revolutionize the working methods of security professionals, thereby unleashing greater productivity. Therefore, further research into the application of LLM agents in cybersecurity is crucial for developing adaptive, intelligent, and comprehensive cybersecurity solutions.

In summary, we bridges the gap between LLM advancements and cybersecurity demands, laying the groundwork for researchers and practitioners. It guides them to harness the transformative potential of LLMs while addressing the unique challenges that arise in this field. Further research and exploration would open up new pathways for future cybersecurity practice, ensuring that we have more comprehensive and professional strategies in the face of increasingly complex cyber threats.

Answer 2 to Q3: Extending the tool-use and API-call capabilities of LLM, coupled with the design of autonomous intelligent agents capable of understanding, planning, and executing complex tasks within cybersecurity applications, will greatly advance the utilization of AI in cybersecurity.

Conclusion

This paper introduces the methodologies for constructing cybersecurity-oriented domain LLMs, detailing how existing models can be fine-tuned to meet specific needs using target data. The investigation into the applications of LLMs has shows that LLMs have great potential for a wide range of cybersecurity tasks, such as threat intelligence, vulnerability detection, secure code generation and others. However, we has also acknowledged the inherent vulnerabilities of LLMs, particularly the susceptibility to attacks such as jailbreaking, which pose significant security risks. Mitigating these vulnerabilities is crucial to securely deploying LLMs in sensitive environments. Additionally, we propose future research directions, such as extending the tool-use and API-call capabilities of LLMs, and developing autonomous intelligent agents for complex cybersecurity operations.

Acknowledgements
We would like to extend our heartfelt gratitude to the editor and the reviewers of this paper. Their professional advice and valuable feedback significantly contributed to the quality of this research.

Author contributions
All authors read and approved the manuscript.

Funding
This work is supported by the Strategic Priority Research Programr of Chinese Academy of Sciences, the System of Cybersecurity Large Model.

Availablity of data and materials
Not applicable.

Declarations

Competing interests
The authors declare that they have no Conflict of interest.

Received: 29 November 2024 Accepted: 8 January 2025
Published online: 05 February 2025

References

Ahmed T, Devanbu P (2023) Better patching using llm prompting, via self-consistency. In: 2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE), pp 1742–1746, https://doi.org/10.1109/ASE56229.2023.00065

Ahmad B, Tan B, Karri R et al (2023a) Flag: finding line anomalies (in code) with generative AI. Preprint at arXiv:2306.12643

Ahmad B, Thakur S, Tan B et al (2023b) Fixing hardware security bugs with large language models. Preprint at arXiv:2302.01215

Alam MT, Bhusal D, Nguyen L et al (2024) CTIBench: a benchmark for evaluating LLMs in cyber threat intelligence. In: The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track

Ali T, Kostakos P (2023) Huntgpt: integrating machine learning-based anomaly detection and explainable AI with large language models (llms). Preprint at arXiv:2309.16021

Almazrouei E, Alobeidli H, Alshamsi A et al (2023) The falcon series of open language models. Preprint at arXiv:2311.16867

Alrashedy K, Aljasser A (2024) Can llms patch security issues? Preprint at arXiv:2312.00024

An K, Yang F, Lu J et al (2024) Nissist: an incident mitigation copilot based on troubleshooting guides. In: Endriss U, Melo FS, Bach K et al (eds) ECAI 2024 - 27th European Conference on Artificial Intelligence, 19-24 October 2024, Santiago de Compostela, Spain - Including 13th Conference on Prestigious Applications of Intelligent Systems (PAIS 2024), Frontiers in Artificial Intelligence and Applications, vol 392. IOS Press, pp 4471–447https://doi.org/10.3233/FAIA241032

Aslan Ö, Aktuğ SS, Ozkan-Okay M et al (2023) A comprehensive review of cyber security vulnerabilities, threats, attacks, and solutions. Electronics 12(6):1333

Asmita, Oliinyk Y, Scott M et al (2024) Fuzzing busybox: leveraging LLM and crash reuse for embedded bug unearthing. In: Balzarotti D, Xu W (eds) 33rd USENIX Security Symposium, USENIX Security 2024, Philadelphia, PA, USA, August 14-16, 2024. USENIX Association, https://www.usenix.org/conference/usenixsecurity24/presentation/asmita

Azizi N, Haass O (2023) Cybersecurity issues and challenges. Handbook of research on cybersecurity issues and challenges for business and FinTech applications. IGI Global, Palmdale, pp 21–48

Bakhshandeh A, Keramatfar A, Norouzi A et al (2023) Using chatgpt as a static application security testing tool. Preprint at arXiv:2308.14434

Barnum S (2012) Standardizing cyber threat intelligence information with the structured threat information expression (stix). Mitre Corp 11:1–22

Barrett C, Boyd B, Bursztein E et al (2023) Identifying and mitigating the security risks of generative AI. Found Trends® Priv Secur 6(1):1–52

Beckerich M, Plein L, Coronado S (2023) Ratgpt: turning online llms into proxies for malware attacks. Preprint at arXiv:2308.09183

Begou N, Vinoy J, Duda A et al (2023) Exploring the dark side of ai: Advanced phishing attack design and deployment using chatgpt. In: 2023 IEEE Conference on Communications and Network Security (CNS), pp 1–6, https://doi.org/10.1109/CNS59707.2023.10288940

Bhatt M, Chennabasappa S, Nikolaidis C et al (2023) Purple llama cyberseceval: A secure coding benchmark for language models. Preprint at arXiv:2312.04724

Bhusal D, Alam MT, Nguyen L et al (2024) Secure: benchmarking generative large language models for cybersecurity advisory. Preprint at arXiv:2405.20441

Bills S, Cammarata N, Mossing D et al (2023) Language models can explain neurons in language models. https://openaipublic.blob.core.windows.net/neuron-explainer/paper/index.html

Bozkurt A, Sharma RC (2023) Generative ai and prompt engineering: The art of whispering to let the genie out of the algorithmic world. Asian Journal of Distance Education 18(2):i–vii

Brown T, Mann B, Ryder N et al (2020) Language models are few-shot learners. Adv Neural Inf Process Syst 33:1877–1901

Buscemi A (2023) A comparative study of code generation using chatgpt 3.5 across 10 programming languages. Preprint at arXiv:2308.04477

Cao T, Huang C, Li Y et al (2024) Phishagent: a robust multimodal agent for phishing webpage detection. Preprint at arXiv:2408.10738

Carlini N, Tramer F, Wallace E et al (2021) Extracting training data from large language models. In: USENIX Security, pp 2633–2650

Carlini N, Jagielski M, Zhang C et al (2022) The privacy onion effect: memorization is relative. In: NeurIPS, http://papers.nips.cc/paper_files/paper/2022/hash/564b5f8289ba846ebc498417e834c253-Abstract-Conference.html

Carlsmith J (2022) Is power-seeking AI an existential risk? Preprint at arXiv:2206.13353

Charalambous Y, Manino E, Cordeiro LC (2024) Automated repair of AI code with large language models and formal verification. Preprint at arXiv:2405.08848

Charan PVS, Chunduri H, Anand PM et al (2023) From text to mitre techniques: Exploring the malicious use of large language models for generating cyber attack payloads. Preprint at arXiv:2305.15336

Chauvin T (2024) eyeballvul: a future-proof benchmark for vulnerability detection in the wild. Preprint at arXiv:2407.08708

Chen T, Li L, Zhu L et al (2023a) Vullibgen: Identifying vulnerable third-party libraries via generative pre-trained model. Preprint at arXiv:2308.04662

Chen Y, Ding Z, Alowain L et al (2023b) Diversevul: a new vulnerable source code dataset for deep learning based vulnerability detection. In: Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses. Association for Computing Machinery, New York, NY, USA, RAID '23, p 654-668, https://doi.org/10.1145/3607199.3607242

Chen X, Lin M, Schärli N et al (2024a) Teaching large language models to self-debug. In: The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024. OpenReview.net, https://openreview.net/forum?id=KuPixIqPiq

Chen Y, Wu J, Ling X et al (2024b) When large language models confront repository-level automatic program repair: how well they done? In: Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings, ICSE Companion 2024, Lisbon, Portugal, April 14-20, 2024. ACM, pp 459–471, https://doi.org/10.1145/3639478.3647633

Cheshkov A, Zadorozhny P, Levichev R (2023) Evaluation of chatgpt model for vulnerability detection. Preprint at arXiv:2304.07232

Chiang WL, Li Z, Lin Z et al (2023) Vicuna: an open-source chatbot impressing gpt-4 with 90%* chatgpt quality. See https://vicuna lmsys org. 2(3):6. Accessed 14 April 2023

Chu J, Liu Y, Yang Z et al (2024) Comprehensive assessment of jailbreak attacks against llms. Preprint at arXiv:2402.05668

Clairoux-Trepanier V, Beauchamp IM, Ruellan E et al (2024) The use of large language models (llm) for cyber threat intelligence (cti) in cybercrime forums. Preprint at arXiv:2408.03354

Clark J, Amodei D (2016) Faulty reward functions in the wild. Internet: https://blogopenaicom/faulty-reward-functions

Cui H, Du Y, Yang Q et al (2024) Llmind: Orchestrating ai and iot with llm for complex task execution. Preprint at arXiv:2312.09007

Dang Y, Huang K, Huo J et al (2024) Explainable and interpretable multimodal large language models: a comprehensive survey. Preprint at arXiv:2412.02104

Das BC, Amini MH, Wu Y (2024) Security and privacy challenges of large language models: a survey. Preprint at arXiv:2402.00888

de Fitero-Dominguez D, García-López E, García-Cabot A (2024) Enhanced automated code vulnerability repair using large language models. Eng Appl Artif Intell 138:109291. https://doi.org/10.1016/J.ENGAPPAI.2024.109291

de Jesus Coelho da Silva G, Westphall CB (2024) A survey of large language models in cybersecurity. Preprint at arXiv:2402.16968

Dehghan M, Wu JJ, Fard FH et al (2024) Mergerepair: An exploratory study on merging task-specific adapters in code llms for automated program repair. Preprint at arXiv:2408.09568

Deng G, Liu Y, Mayoral-Vilches V et al (2023a) Pentestgpt: An llm-empowered automatic penetration testing tool. Preprint at arXiv:2308.06782

Deng Y, Xia CS, Peng H et al (2023b) Large language models are zero-shot fuzzers: fuzzing deep-learning libraries via large language models. In: Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis. Association for Computing Machinery, New York, NY, USA, ISSTA 2023, p 423-435, https://doi.org/10.1145/3597926.3598067

Deng Y, Xia CS, Yang C et al (2023c) Large language models are edge-case fuzzers: Testing deep learning libraries via fuzzgpt. Preprint at arXiv:2304.02014

Deng G, Liu Y, Li Y et al (2024) Masterkey: automated jailbreaking of large language model chatbots. In: Proceedings 2024 Network and Distributed System Security Symposium. Internet Society, NDSS 2024, https://doi.org/10.14722/ndss.2024.24188

Dettmers T, Pagnoni A, Holtzman A et al (2024) Qlora: efficient finetuning of quantized llms. Adv Neural Inf Process Syst. 36

Dev S, Jha A, Goyal J et al (2023) Building stereotype repositories with llms and community engagement for scale and depth. Cross-Cultural Considerations in NLP@ EACL 84

Ding N, Qin Y, Yang G et al (2023) Parameter-efficient fine-tuning of large-scale pre-trained language models. Nat Machi Intell 5(3):220–235

Ding P, Kuang J, Ma D et al (2024) A wolf in sheep's clothing: Generalized nested jailbreak prompts can fool large language models easily. In: Duh K, Gómez-Adorno H, Bethard S (eds) Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024. Association for Computational Linguistics, pp 2136–2153, https://doi.org/10.18653/V1/2024.NAACL-LONG.118

Donadel D, Marchiori F, Pajola L et al (2024) Can llms understand computer networks? towards a virtual system administrator. Preprint at arXiv:2404.12689

Dong G, Yuan H, Lu K et al (2023) How abilities in large language models are affected by supervised fine-tuning data composition. Preprint at arXiv:2310.05492

Dong X, Wang Y, Yu PS et al (2024) Disclosure and mitigation of gender bias in llms. Preprint at arXiv:2402.11190

Du X, Wen M, Zhu J et al (2024a) Generalization-enhanced code vulnerability detection via multi-task instruction fine-tuning. In: Ku L, Martins A, Srikumar V (eds) Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024. Association for Computational Linguistics, pp 10507–10521, https://doi.org/10.18653/V1/2024.FINDINGS-ACL.625

Du X, Zheng G, Wang K et al (2024b) Vul-rag: Enhancing llm-based vulnerability detection via knowledge-level rag. Preprint at arXiv:2406.11147

Elhage N, Nanda N, Olsson C et al (2021) A mathematical framework for transformer circuits. Trans Circuits Thread 1(1):12

Elhage N, Hume T, Olsson C et al (2022) Toy models of superposition. Trans Circuits Thread

Esmradi A, Yip DW, Chan C (2023) A comprehensive survey of attack techniques, implementation, and mitigation strategies in large language models. In: Wang G, Wang H, Min G et al (eds) Ubiquitous Security - Third International Conference, UbiSec 2023, Exeter, UK, November 1-3, 2023, Revised Selected Papers, Communications in Computer and Information Science, vol 2034. Springer, pp 76–95, https://doi.org/10.1007/978-981-97-1274-8_6

Fang C, Miao N, Srivastav S et al (2024a) Large language models for code analysis: Do llms really do their job? In: Balzarotti D, Xu W (eds) 33rd USENIX Security Symposium, USENIX Security 2024, Philadelphia, PA, USA, August 14-16, 2024. USENIX Association, https://www.usenix.org/conference/usenixsecurity24/presentation/fang

Fang R, Bindu R, Gupta A et al (2024b) Llm agents can autonomously exploit one-day vulnerabilities. Preprint at arXiv:2404.08144

Fang R, Bindu R, Gupta A et al (2024c) Llm agents can autonomously hack websites. Preprint at arXiv:2402.06664

Fang R, Bindu R, Gupta A et al (2024d) Teams of llm agents can exploit zero-day vulnerabilities. Preprint at arXiv:2406.01637

Fayyazi R, Yang SJ (2023) On the uses of large language models to interpret ambiguous cyberattack descriptions. Preprint at arXiv:2306.14062

Fayyazi R, Taghdimi R, Yang SJ (2024) Advancing ttp analysis: Harnessing the power of encoder-only and decoder-only language models with retrieval augmented generation. Preprint at arXiv:2401.00280

Feng S, Chen C (2024) Prompting is all you need: Automated android bug replay with large language models. In: Proceedings of the 46th IEEE/ACM International Conference on Software Engineering, ICSE 2024, Lisbon, Portugal, April 14-20, 2024. ACM, pp 67:1–67:13,https://doi.org/10.1145/3597503.3608137

Ferrag MA, Battah A, Tihanyi N et al (2023) Securefalcon: The next cyber reasoning system for cyber security. Preprint at arXiv:2307.06616

Ferrag MA, Alwahedi F, Battah A et al (2024a) Generative ai and large language models for cyber security: All insights you need. Preprint at arXiv:2405.12750

Ferrag MA, Ndhlovu M, Tihanyi N et al (2024b) Revolutionizing cyber threat detection with large language models: a privacy-preserving bert-based lightweight model for iot/iiot devices. IEEE Access 12:23733–23750. https://doi.org/10.1109/ACCESS.2024.3363469

Fieblinger R, Alam MT, Rastogi N (2024) Actionable cyber threat intelligence using knowledge graphs and large language models. In: 2024 IEEE European Symposium on Security and Privacy Workshops (EuroS &PW), IEEE, pp 100–111

Francia J, Hansen D, Schooley B et al (2024) Assessing AI vs human-authored spear phishing sms attacks: An empirical study using the trapd method. Preprint at arXiv:2406.13049

Fujima H, Kumamoto T, Yoshida Y (2023) Using chatgpt to analyze ransomware messages and to predict ransomware threats. https://doi.org/10.21203/rs.3.rs-3645967/v1

Gao L, la Tour TD, Tillman H et al (2024) Scaling and evaluating sparse autoencoders. Preprint at arXiv:2406.04093

Gao Z, Wang H, Zhou Y et al (2023) How far have we gone in vulnerability detection using large language models. Preprint at arXiv:2311.12420

Ge Y, Hua W, Mei K et al (2024) Openagi: When llm meets domain experts. Adv Neural Inf Process Syst. 36

Ghelani D (2022) Cyber security, cyber threats, implications and future perspectives: a review. Authorea Preprints

Gonçalves J, Dias T, Maia E et al (2024) Scope: Evaluating llms for software vulnerability detection. Preprint at arXiv:2407.14372

Guastalla M, Li Y, Hekmati A et al (2023) Application of large language models to ddos attack detection. In: International Conference on Security and Privacy in Cyber-Physical Systems and Smart Vehicles, Springer, pp 83–99

Gulati A, Qin J, Chiu C et al (2020) Conformer: Convolution-augmented transformer for speech recognition. In: Meng H, Xu B, Zheng TF (eds) 21st Annual Conference of the International Speech Communication Association, Interspeech 2020, Virtual Event, Shanghai, China, October 25-29, 2020. ISCA, pp 5036–5040, https://doi.org/10.21437/INTERSPEECH.2020-3015

Guo H, Yang J, Liu J et al (2024a) OWL: A large language model for IT operations. In: The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024. OpenReview.net, https://openreview.net/forum?id=SZOQ9RKYJu

Guo Y, Patsakis C, Hu Q et al (2024b) Outside the comfort zone: Analysing llm capabilities in software vulnerability detection. In: European symposium on research in computer security, Springer, pp 271–289

Gupta M, Akiri C, Aryal K et al (2023) From chatgpt to threatgpt: impact of generative AI in cybersecurity and privacy. IEEE Access 11:80218–80245. https://doi.org/10.1109/ACCESS.2023.3300381

Han X, Yuan S, Trabelsi M (2023) Loggpt: Log anomaly detection via GPT. In: He J, Palpanas T, Hu X et al (eds) IEEE International Conference on Big Data, BigData 2023, Sorrento, Italy, December 15-18, 2023. IEEE, pp 1117–1122, https://doi.org/10.1109/BIGDATA59044.2023.10386543

Happe A, Cito J (2023) Getting pwn'd by AI: Penetration testing with large language models. In: Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. Association for Computing Machinery, New York, NY, USA, ESEC/FSE 2023, p 2082-2086, https://doi.org/10.1145/3611643.3613083

Happe A, Kaplan A, Cito J (2023) Evaluating llms for privilege-escalation scenarios. Preprint at arXiv:2310.11409

Hays S, White DJ (2024) Employing llms for incident response planning and review. Preprint at arXiv:2403.01271

He J, Vechev M (2023) Large language models for code: Security hardening and adversarial testing. In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security. ACM, CCS '23, https://doi.org/10.1145/3576915.3623175

He J, Vero M, Krasnopolska G et al (2024) Instruction tuning for secure code generation. In: Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024. OpenReview.net, https://openreview.net/forum?id=MgTzMaYHvG

He R, Liu L, Ye H et al (2021) On the effectiveness of adapter-based tuning for pretrained language model adaptation. In: Zong C, Xia F, Li W et al (eds) Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021. Association for Computational Linguistics, pp 2208–2222, https://doi.org/10.18653/V1/2021.ACL-LONG.172

Heiding F, Schneier B, Vishwanath A et al (2023) Devising and detecting phishing: Large language models vs. smaller human models. Preprint at arXiv:2308.12287

Hendrycks D, Mazeika M, Woodside T (2023) An overview of catastrophic AI risks. Preprint at arXiv:2306.12001

Hossen MI, Zhang J, Cao Y et al (2024) Assessing cybersecurity vulnerabilities in code large language models. Preprint at arXiv:2404.18567
Hou X, Zhao Y, Liu Y et al (2023) Large language models for software engineering: a systematic literature review. ACM Trans Softw Eng Methodol
Hu EJ, Shen Y, Wallis P et al (2022) Lora: Low-rank adaptation of large language models. In: The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022. OpenReview.net, https://openreview.net/forum?id=nZeVKeeFYf9
Hu J, Zhang Q, Yin H (2023a) Augmenting greybox fuzzing with generative AI. Preprint at arXiv:2306.06782
Hu S, Huang T, Ilhan F et al (2023b) Large language model-powered smart contract vulnerability detection: New perspectives. In: 5th IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications, TPS-ISA 2023, Atlanta, GA, USA, November 1-4, 2023. IEEE, pp 297–306, https://doi.org/10.1109/TPS-ISA58951.2023.00044
Hu Y, Zou F, Han J et al (2024) LLM-TIKG: threat intelligence knowledge graph construction utilizing large language model. Comput Secur 145:103999. https://doi.org/10.1016/J.COSE.2024.103999
Huang J, Zhu Q (2023) Penheal: a two-stage llm framework for automated pentesting and optimal remediation. In: Proceedings of the Workshop on Autonomous Cybersecurity, pp 11–22
Huang L, Yu W, Ma W et al (2023a) A survey on hallucination in large language models: principles, taxonomy, challenges, and open questions. ACM Trans Inf Syst
Huang Y, Du H, Zhang X et al (2023b) Large language models for networking: Applications, enabling techniques, and challenges. Preprint at arXiv:2311.17474
Huang Y, Sun L, Wang H et al (2024) Trustllm: Trustworthiness in large language models. In: Forty-first International Conference on Machine Learning, ICML 2024,Vienna, Austria, July 21-27, 2024
Huben R, Cunningham H, Smith LR et al (2023) Sparse autoencoders find highly interpretable features in language models. In: The Twelfth International Conference on Learning Representations
Hubinger E, Denison C, Mu J et al (2024) Sleeper agents: Training deceptive llms that persist through safety training. Preprint at arXiv:2401.05566
Ibrahim A, Thérien B, Gupta K et al (2024) Simple and scalable strategies to continually pre-train large language models. Preprint at arXiv:2403.08763
Inan H, Upasani K, Chi J et al (2023) Llama guard: Llm-based input-output safeguard for human-AI conversations. Preprint at arXiv:2312.06674
Ince P, Luo X, Yu J et al (2024) Detect Llama - Finding Vulnerabilities in Smart Contracts Using Large Language Models, Springer, Singapore, p 424-443. https://doi.org/10.1007/978-981-97-5101-3_23
Islam NT, Khoury J, Seong A et al (2024) Llm-powered code vulnerability repair with reinforcement learning and semantic reward. Preprint at arXiv:2401.03374
Jamal S, Wimmer H (2023) An improved transformer-based model for detecting phishing, spam, and ham: a large language model approach. Preprint at arXiv:2311.04913
Jensen RIT, Tawosi V, Alamir S (2024) Software vulnerability and functionality assessment using llms. Preprint at arXiv:2403.08429
Ji H, Yang J, Chai L et al (2024) Sevenllm: Benchmarking, eliciting, and enhancing abilities of large language models in cyber threat intelligence. Preprint at arXiv:2405.03446
Ji Z, Lee N, Frieske R et al (2023) Survey of hallucination in natural language generation. ACM Comput Surv 55(12):1–38
Jiang N, Wang C, Liu K et al (2023a) Nova$^+$: Generative language models for binaries. Preprint at arXiv:2311.13721
Jiang S, Chen X, Tang R (2023b) Prompt packer: deceiving llms through compositional instruction with hidden attacks. Preprint at arXiv:2310.10077
Jiang AQ, Sablayrolles A, Roux A et al (2024a) Mixtral of experts. Preprint at arXiv:2401.04088
Jiang Y, Liang J, Ma F et al (2024b) When fuzzing meets llms: Challenges and opportunities. In: Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering, pp 492–496
Jin J, Tang B, Ma M et al (2024) Crimson: Empowering strategic reasoning in cybersecurity through large language models. Preprint at arXiv:2403.00878
Jin M, Shahriar S, Tufano M et al (2023) Inferfix: end-to-end program repair with llms. In: Chandra S, Blincoe K, Tonella P (eds) Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, ESEC/FSE 2023, San Francisco, CA, USA, December 3-9, 2023. ACM, pp 1646–1656https://doi.org/10.1145/3611643.3613892
Kabir S, Udo-Imeh DN, Kou B et al (2024) Is stack overflow obsolete? an empirical study of the characteristics of chatgpt answers to stack overflow questions. In: Mueller FF, Kyburz P, Williamson JR et al (eds) Proceedings of the CHI Conference on Human Factors in Computing Systems, CHI 2024, Honolulu, HI, USA, May 11-16, 2024. ACM, pp 935:1–935:17, https://doi.org/10.1145/3613904.3642596
Kaheh M, Kholgh DK, Kostakos P (2023) Cyber sentinel: Exploring conversational agents in streamlining security tasks with gpt-4. Preprint at arXiv:2309.16422
Karlsen E, Luo X, Zincir-Heywood N et al (2024) Benchmarking large language models for log analysis, security, and interpretation. J Netw Syst Manag 32(3):59. https://doi.org/10.1007/S10922-024-09831-X
Kaur P, Kashyap GS, Kumar A et al (2024) From text to transformation: A comprehensive review of large language models' versatility. Preprint at arXiv:2402.16142
Kaur R, Gabrijelčič D, Klobučar T (2023) Artificial intelligence for cybersecurity: Literature review and future research directions. Information Fusion NA:101804
Kavian A, Pourhashem Kallehbasti MM, Kazemi S et al (2024) Llm security guard for code. In: Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering, pp 600–603
Keller J, Nowakowski J (2024) Ai-powered patching: the future of automated vulnerability fixes. Tech. rep., google
Khare A, Dutta S, Li Z et al (2023) Understanding the effectiveness of large language models in detecting security vulnerabilities. Preprint at arXiv:2311.16169
Khoury R, Avila AR, Brunelle J et al (2023) How secure is code generated by chatgpt? In: IEEE International Conference on Systems, Man, and Cybernetics, SMC 2023, Honolulu, Oahu, HI, USA, October 1-4, 2023. IEEE, pp 2445–2451, https://doi.org/10.1109/SMC53992.2023.10394237
Kong J, Cheng M, Xie X et al (2024) Contrastrepair: Enhancing conversation-based automated program repair via contrastive test case pairs. Preprint at arXiv:2403.01971
Kouliaridis V, Karopoulos G, Kambourakis G (2024) Assessing the effectiveness of llms in android application vulnerability analysis. Preprint at arXiv:2406.18894
Kour G, Zalmanovici M, Zwerdling N et al (2023) Unveiling safety vulnerabilities of large language models. Preprint at arXiv:2311.04124
Kulsum U, Zhu H, Xu B et al (2024) A case study of llm for automated vulnerability repair: Assessing impact of reasoning and patch validation feedback. In: Proceedings of the 1st ACM International Conference on AI-Powered Software, pp 103–111
Kumar S, Gupta U, Singh AK et al (2023) Artificial intelligence: revolutionizing cyber security in the digital era. J Comput Mech Manag 2(3):31–42
Kumar SS, Cummings M, Stimpson A (2024) Strengthening llm trust boundaries: A survey of prompt injection attacks surender suresh kumar dr. ml cummings dr. alexander stimpson. In: 2024 IEEE 4th International Conference on Human-Machine Systems (ICHMS), IEEE, pp 1–6
Lai J, Gan W, Wu J et al (2024) Large language models in law: a survey. AI Open. 5:181
Lapid R, Langberg R, Sipper M (2023) Open sesame! universal black box jailbreaking of large language models. Preprint at arXiv:2309.01446
Le TK, Alimadadi S, Ko SY (2024) A study of vulnerability repair in javascript programs with large language models. In: Chua T, Ngo C, Lee RK et al (eds) Companion Proceedings of the ACM on Web Conference 2024, WWW 2024, Singapore, Singapore, May 13-17, 2024. ACM, pp 666–669, https://doi.org/10.1145/3589335.3651463
Lee YT, Vijayakumar H, Qian Z et al (2024) Static detection of filesystem vulnerabilities in android systems. Preprint at arXiv:2407.11279
Lemieux C, Inala JP, Lahiri SK et al (2023) Codamosa: Escaping coverage plateaus in test generation · with pre-trained large language models. In: 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), pp 919–931, https://doi.org/10.1109/ICSE48619.2023.00085
Lester B, Al-Rfou R, Constant N (2021) The power of scale for parameter-efficient prompt tuning. In: Moens M, Huang X, Specia L et al (eds) Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana,

Dominican Republic, 7-11 November, 2021. Association for Computational Linguistics, pp 3045–3059, https://doi.org/10.18653/V1/2021.EMNLP-MAIN.243

Levi M, Alluouche Y, Ohayon D et al (2024) Cyberpal. ai: Empowering llms with expert-driven cybersecurity instructions. Preprint at arXiv:2408.09304

Li G, Li Y, Guannan W et al (2023a) Seceval: A comprehensive benchmark for evaluating cybersecurity knowledge of foundation models. https://github.com/XuanwuAI/SecEval

Li H, Guo D, Fan W et al (2023b) Multi-step jailbreaking privacy attacks on chatgpt. In: Bouamor H, Pino J, Bali K (eds) Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023. Association for Computational Linguistics, pp 4138–4153, https://doi.org/10.18653/V1/2023.FINDINGS-EMNLP.272

Li H, Hao Y, Zhai Y et al (2023c) The hitchhiker's guide to program analysis: a journey with large language models. Preprint at arXiv:2308.00245

Li R, Allal LB, Zi Y et al (2023d) Starcoder: may the source be with you! Trans Mach Learn Res 2023. https://openreview.net/forum?id=KoFOg41haE

Li Y, Du M, Song R et al (2023e) A survey on fairness in large language models. Preprint at arXiv:2308.10149

Li Y, Wang S, Ding H et al (2023f) Large language models in finance: a survey. In: Proceedings of the Fourth ACM International Conference on AI in Finance, pp 374–382

Li G, Zhi C, Chen J et al (2024a) Exploring parameter-efficient fine-tuning of large language model on automated program repair. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering, pp 719–731

Li J, Rabbi F, Cheng C et al (2024b) An exploratory study on fine-tuning large language models for secure code generation. Preprint at arXiv:2408.09078

Li N, Pan A, Gopal A et al (2024c) The WMDP benchmark: Measuring and reducing malicious use with unlearning. In: Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024. OpenReview.net

Li Z, Dutta S, Naik M (2024d) Llm-assisted static analysis for detecting security vulnerabilities. Preprint at arXiv:2405.17238

Li Y, Liu Q (2021) A comprehensive review study of cyber-attacks and cyber security; emerging trends and recent developments. Energy Rep 7:8176–8186

Lin YZ, Mamun M, Chowdhury MA et al (2023) Hw-v2w-map: Hardware vulnerability to weakness mapping framework for root cause analysis with gpt-assisted mitigation suggestion. Preprint at arXiv:2312.13530

Lin Z, Cui J, Liao X et al (2024) Malla: Demystifying real-world large language model integrated malicious services. In: Balzarotti D, Xu W (eds) 33rd USENIX Security Symposium, USENIX Security 2024, Philadelphia, PA, USA, August 14-16, 2024. USENIX Association, https://www.usenix.org/conference/usenixsecurity24/presentation/lin-zilong

Liu B, Huo W, Zhang C et al (2018) αdiff: cross-version binary code similarity detection with dnn. In: Proceedings of the 33rd ACM/IEEE international conference on automated software engineering, pp 667–678

Liu H, Wang Y, Fan W et al (2023a) Trustworthy ai: A computational perspective. ACM Transactions on Intelligent Systems and Technology p 1-59. https://doi.org/10.1145/3546872, http://dx.doi.org/10.1145/3546872

Liu J, Huang J, Huo Y et al (2023b) Log-based anomaly detection based on evt theory with feedback. Preprint at arXiv:2306.05032

Liu P, Liu J, Fu L et al (2023c) How chatgpt is solving vulnerability management problem. Preprint at arXiv:2311.06530

Liu P, Sun C, Zheng Y et al (2023d) Harnessing the power of llm to support binary taint analysis. Preprint at arXiv:2310.08275

Liu X, Zheng Y, Du Z et al (2023e) Gpt understands, too. AI Open. NA

Liu Y, Jia Y, Geng R et al (2023f) Prompt injection attacks and defenses in llm-integrated applications. Preprint at arXiv:2310.12815

Liu X, Ji K, Fu Y et al (2021) P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks. Preprint at arXiv:2110.07602

Liu Z (2023) Secqa: A concise question-answering dataset for evaluating large language models in computer security. Preprint at arXiv:2312.15838

Liu J, Xia CS, Wang Y et al (2024a) Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. Adv Neural Inf Process Syst. 36

Liu Y, Pei C, Xu L et al (2024b) Opseval: A comprehensive it operations benchmark suite for large language models. Preprint at arXiv:2310.07637

Liu Y, Tao S, Meng W et al (2024c) Interpretable online log analysis using large language models with prompt strategies. In: Steinmacher I, Linares-Vásquez M, Moran KP et al (eds) Proceedings of the 32nd IEEE/ACM International Conference on Program Comprehension, ICPC 2024, Lisbon, Portugal, April 15-16, 2024. ACM, pp 35–46, https://doi.org/10.1145/3643916.3644408

Liu Y, Yuan Y, Wang C et al (2024d) From summary to action: Enhancing large language models for complex tasks with open world apis. Preprint at arXiv:2402.18157

Liu Z, Chen C, Wang J et al (2024e) Make LLM a testing expert: Bringing human-like interaction to mobile GUI testing via functionality-aware decisions. In: Proceedings of the 46th IEEE/ACM International Conference on Software Engineering, ICSE 2024, Lisbon, Portugal, April 14-20, 2024. ACM, pp 100:1–100:13, https://doi.org/10.1145/3597503.3639180

Liu Z, Shi J, Buford JF (2024f) Cyberbench: A multi-task benchmark for evaluating large language models in cybersecurity. In: AAAI 2024 Workshop on Artificial Intelligence for Cyber Security

Liu Z, Liao Q, Gu W et al (2023g) Software vulnerability detection with gpt and in-context learning. In: 2023 8th International Conference on Data Science in Cyberspace (DSC), pp 229–236, https://doi.org/10.1109/DSC59305.2023.00041

Liu Z, Tang Y, Luo X et al (2024) No need to lift a finger anymore? assessing the quality of code generation by chatgpt. IEEE Trans Software Eng 50(6):1548–1584. https://doi.org/10.1109/TSE.2024.3392499

Lozhkov A, Li R, Allal LB et al (2024) Starcoder 2 and the stack v2: The next generation. Preprint at arXiv:2402.19173

Lu H, Peng H, Nan G et al (2024) Malsight: Exploring malicious source code and benign pseudocode for iterative binary malware summarization. Preprint at arXiv:2406.18379

Luo Z, Xu C, Zhao P et al (2024) Wizardcoder: Empowering code large language models with evol-instruct. In: The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024. OpenReview.net, https://openreview.net/forum?id=UnUwSIgK5W

Mahyari AA (2024) Harnessing the power of llms in source code vulnerability detection. Preprint at arXiv:2408.03489

Mani SK, Zhou Y, Hsieh K et al (2023) Enhancing network management using code generated by large language models. In: Proceedings of the 22nd ACM Workshop on Hot Topics in Networks. Association for Computing Machinery, New York, NY, USA, HotNets '23, p 196-204, https://doi.org/10.1145/3626111.3628183

Mao Q, Li Z, Hu X et al (2024a) Towards effectively detecting and explaining vulnerabilities using large language models. Preprint at arXiv:2406.09701

Mao Z, Li J, Li M et al (2024b) Multi-role consensus through llms discussions for vulnerability detection. Preprint at arXiv:2403.14274

Mathews NS, Brus Y, Aafer Y et al (2024) Llbezpeky: Leveraging large language models for vulnerability detection. Preprint at arXiv:2401.01269

Maynez J, Narayan S, Bohnet B et al (2020) On faithfulness and factuality in abstractive summarization. In: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics, Online, pp 1906–1919, https://doi.org/10.18653/v1/2020.acl-main.173

Meng R, Mirchev M, Böhme M et al (2024) Large language model guided protocol fuzzing. In: 31st Annual Network and Distributed System Security Symposium, NDSS 2024, San Diego, California, USA, February 26 - March 1, 2024. The Internet Society, https://www.ndss-symposium.org/ndss-paper/large-language-model-guided-protocol-fuzzing/

Miao Y, Bai Y, Chen L et al (2023) An empirical study of netops capability of pre-trained large language models. Preprint at arXiv:2309.05557

Michelet G, Breitinger F (2024) Chatgpt, llama, can you write my report? an experiment on assisted digital forensics reports written using (local) large language models. Forensic Sci Int Digit Investig 48:301683. https://doi.org/10.1016/J.FSIDI.2023.301683

Mijwil M, Aljanabi M et al (2023) Towards artificial intelligence-based cybersecurity: the practices and chatgpt generated ways to combat cybercrime. Iraqi J Comput Sci Math 4(1):65–70

Minaee S, Mikolov T, Nikzad N et al (2024) Large language models: a survey. Preprint at arXiv:2402.06196

Mitra S, Neupane S, Chakraborty T et al (2024) Localintel: Generating organizational threat intelligence from global and local cyber knowledge. Preprint at arXiv:2401.10036
Mohammed SP, Hossain G (2024) Chatgpt in education, healthcare, and cybersecurity: Opportunities and challenges. In: 2024 IEEE 14th Annual Computing and Communication Workshop and Conference (CCWC), IEEE, pp 0316–0321
Moskal S, Laney S, Hemberg E et al (2023) Llms killed the script kiddie: How agents supported by large language models change the landscape of network threat testing. Preprint at arXiv:2310.06936
Motlagh FN, Hajizadeh M, Majd M et al (2024) Large language models in cybersecurity: state-of-the-art. Preprint at arXiv:2402.00891
Mtsweni J, Gcaza N, Thaba M (2018) A unified cybersecurity framework for complex environments. In: Proceedings of the Annual Conference of the South African Institute of Computer Scientists and Information Technologists, pp 1–9
Murthy S, Bakar AA, Rahim FA et al (2019) A comparative study of data anonymization techniques. In: HPSC, IEEE, pp 306–309
Nagy B, Hegedűs I, Sándor N et al (2023) Privacy-preserving federated learning and its application to natural language processing. Knowledge-Based Syst 268:110475
Nahmias D, Engelberg G, Klein D et al (2024) Prompted contextual vectors for spear-phishing detection. Preprint at arXiv:2402.08309
Nijkamp E, Hayashi H, Xiong C et al (2023a) Codegen2: Lessons for training llms on programming and natural languages. Preprint at arXiv:2305.02309
Nijkamp E, Pang B, Hayashi H et al (2023b) Codegen: An open large language model for code with multi-turn program synthesis. In: The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net, https://openreview.net/forum?id=iaYcJKpY2B_
Omar M, Shiaeles S (2023) Vuldetect: A novel technique for detecting software vulnerabilities using language models. In: IEEE International Conference on Cyber Security and Resilience, CSR 2023, Venice, Italy, July 31 - Aug. 2, 2023. IEEE, pp 105–110, https://doi.org/10.1109/CSR57506.2023.10224924
Ouyang L, Wu J, Jiang X et al (2022) Training language models to follow instructions with human feedback. Adv Neural Inf Process Syst 35:27730–27744
Palacio DN, Velasco A, Rodriguez-Cardenas D et al (2023) Evaluating and explaining large language models for code using syntactic structures. Preprint at arXiv:2308.03873
Pankajakshan R, Biswal S, Govindarajulu Y et al (2024) Mapping llm security landscapes: a comprehensive stakeholder risk assessment proposal. Preprint at arXiv:2403.13309
Paria S, Dasgupta A, Bhunia S (2023) Divas: An llm-based end-to-end framework for soc security analysis and policy-based protection. Preprint at arXiv:2308.06932
Pasupuleti R, Vadapalli R, Mader C (2023) Cyber security issues and challenges related to generative ai and chatgpt. In: 2023 Tenth International Conference on Social Networks Analysis, Management and Security (SNAMS), pp 1–5, https://doi.org/10.1109/SNAMS60348.2023.10375472
Pearce H, Ahmad B, Tan B et al (2022a) Asleep at the keyboard? assessing the security of github copilot's code contributions. In: 2022 IEEE Symposium on Security and Privacy (SP), pp 754–768, https://doi.org/10.1109/SP46214.2022.9833571
Pearce H, Tan B, Krishnamurthy P et al (2022b) Pop quiz! can a large language model help with reverse engineering? Preprint at arXiv:2202.01142
Pearce H, Tan B, Ahmad B et al (2023) Examining zero-shot vulnerability repair with large language models. In: 2023 IEEE Symposium on Security and Privacy (SP), pp 2339–2356, https://doi.org/10.1109/SP46215.2023.10179324
Pedro R, Castro D, Carreira P et al (2023) From prompt injections to sql injection attacks: How protected is your llm-integrated web application? Preprint at arXiv:2308.01990
Peris C, Dupuy C, Majmudar J et al (2023) Privacy in the time of language models. In: WSDM, pp 1291–1292
Perrina F, Marchiori F, Conti M et al (2023) AGIR: automating cyber threat intelligence reporting with natural language generation. In: He J, Palpanas T, Hu X et al (eds) IEEE International Conference on Big Data, BigData 2023, Sorrento, Italy, December 15-18, 2023. IEEE, pp 3053–3062, https://doi.org/10.1109/BIGDATA59044.2023.10386116
Piet J, Alrashed M, Sitawarin C et al (2024) Jatmo: Prompt injection defense by task-specific finetuning. In: García-Alfaro J, Kozik R, Choras M et al (eds) Computer Security - ESORICS 2024 - 29th European Symposium on Research in Computer Security, Bydgoszcz, Poland, September 16-20, 2024, Proceedings, Part I, Lecture Notes in Computer Science, vol 14982. Springer, pp 105–124, https://doi.org/10.1007/978-3-031-70879-4_6
Prapty RT, Kundu A, Iyengar A (2024) Using retriever augmented large language models for attack graph generation. Preprint at arXiv:2408.05855
Pratama D, Suryanto N, Adiputra AA et al (2024) Cipher: Cybersecurity intelligent penetration-testing helper for ethical researcher. Sensors 24(21):6878. https://doi.org/10.3390/s24216878
Prenner JA, Robbes R (2021) Automatic program repair with openai's codex: Evaluating quixbugs. Preprint at arXiv:2111.03922
Purba MD, Ghosh A, Radford BJ et al (2023) Software vulnerability detection using large language models. In: 2023 IEEE 34th International Symposium on Software Reliability Engineering Workshops (ISSREW), pp 112–119, https://doi.org/10.1109/ISSREW60843.2023.00058
Qi J, Huang S, Luan Z et al (2023) Loggpt: Exploring chatgpt for log-based anomaly detection. In: IEEE International Conference on High Performance Computing & Communications, Data Science & Systems, Smart City & Dependability in Sensor, Cloud & Big Data Systems & Application, HPCC/DSS/SmartCity/DependSys 2023, Melbourne, Australia, December 17-21, 2023. IEEE, pp 273–280, https://doi.org/10.1109/HPCC-DSS-SMARTCITY-DEPENDSYS60770.2023.00045
Qiao B, Li L, Zhang X et al (2023) Taskweaver: A code-first agent framework. Preprint at arXiv:2311.17541
Qin Y, Liang S, Ye Y et al (2024) Toolllm: Facilitating large language models to master 16000+ real-world apis. In: The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024. OpenReview.net, https://openreview.net/forum?id=dHng2O0Jjr
Qiu H, Zhang S, Li A et al (2023) Latent jailbreak: A benchmark for evaluating text safety and output robustness of large language models. Preprint at arXiv:2307.08487
Radford A, Narasimhan K (2018) Improving language understanding by generative pre-training. https://api.semanticscholar.org/CorpusID:49313245
Rajapaksha S, Rani R, Karafili E (2024) A rag-based question-answering solution for cyber-attack investigation and attribution. Preprint at arXiv:2408.06272
Reti D, Becker N, Angeli T et al (2024) Act as a honeytoken generator! an investigation into honeytoken generation with large language models. Preprint at arXiv:2404.16118
Rigaki M, Lukás O, Catania CA et al (2024) Out of the cage: How stochastic parrots win in cyber security environments. In: Rocha AP, Steels L, van den Herik HJ (eds) Proceedings of the 16th International Conference on Agents and Artificial Intelligence, ICAART 2024, Volume 3, Rome, Italy, February 24-26, 2024. SCITEPRESS, pp 774–781, https://doi.org/10.5220/0012391800003636
Roy SS, Thota P, Naragam KV et al (2024) From chatbots to phishbots? – preventing phishing scams created using chatgpt, google bard and claude. Preprint at arXiv:2310.19181
Roziere B, Gehring J, Gloeckle F et al (2023) Code llama: Open foundation models for code. Preprint at arXiv:2308.12950
Ruan J, Chen Y, Zhang B et al (2023) Tptu: Large language model-based ai agents for task planning and tool usage. Preprint at arXiv:2308.03427
Saha D, Tarek S, Yahyaei K et al (2024) LLM for soc security: a paradigm shift. IEEE Access 12:155498–155521. https://doi.org/10.1109/ACCESS.2024.3427369
Sahoo P, Singh AK, Saha S et al (2024) A systematic survey of prompt engineering in large language models: Techniques and applications. Preprint at arXiv:2402.07927
Sandoval G, Pearce H, Nys T et al (2023) Lost at c: A user study on the security implications of large language model code assistants. In: 32nd USENIX Security Symposium (USENIX Security 23). USENIX Association, Anaheim, CA, pp 2205–2222, https://www.usenix.org/conference/usenixsecurity23/presentation/sandoval
Sarabi A, Yin T, Liu M (2023) An llm-based framework for fingerprinting internet-connected devices. In: Proceedings of the 2023 ACM on Internet Measurement Conference. Association for Computing Machinery,

New York, NY, USA, IMC '23, p 478-484, https://doi.org/10.1145/3618257.3624845
Scala NM, Reilly AC, Goethals PL et al (2019) Risk and the five hard problems of cybersecurity. Risk Anal 39(10):2119–2126
Scanlon M, Breitinger F, Hargreaves C et al (2023) Chatgpt for digital forensic investigation: the good, the bad, and the unknown. Forensic Sci Int Digit Investig. 46:301609. https://doi.org/10.1016/j.fsidi.2023.301609 (https://www.sciencedirect.com/science/article/pii/S266628172300121X)
Schwartz Y, Benshimol L, Mimran D et al (2024) Llmcloudhunter: Harnessing llms for automated extraction of detection rules from cloud-based cti. Preprint at arXiv:2407.05194
Sekhari A, Acharya J, Kamath G et al (2021) Remember what you want to forget: Algorithms for machine unlearning. In: NeurIPS, pp 18075–18086
Shao M, Jancheska S, Udeshi M et al (2024) Nyu ctf dataset: A scalable open-source benchmark dataset for evaluating llms in offensive security. Preprint at arXiv:2406.05590
Sharma P, Dash B (2023) Impact of big data analytics and chatgpt on cybersecurity. In: 2023 4th International Conference on Computing and Communication Systems (I3CS), pp 1–6, https://doi.org/10.1109/I3CS58314.2023.10127411
Shen X, Chen Z, Backes M et al (2023) "do anything now": Characterizing and evaluating in-the-wild jailbreak prompts on large language models. Preprint at arXiv:2308.03825
Shestov A, Levichev R, Mussabayev R et al (2024) Finetuning large language models for vulnerability detection. Preprint at arXiv:2401.17010
Shi J, Liu Y, Zhou P et al (2023) Badgpt: Exploring security vulnerabilities of chatgpt via backdoor attacks to instructgpt. Preprint at arXiv:2304.12298
Siddiq ML, Santos JCS (2022) Securityeval dataset: mining vulnerability examples to evaluate machine learning-based code generation techniques. In: Proceedings of the 1st International Workshop on Mining Software Repositories Applications for Privacy and Security. Association for Computing Machinery, New York, NY, USA, MSR4P &S 2022, p 29-33, https://doi.org/10.1145/3549035.3561184
Siddiq ML, Santos JCS (2023) Generate and pray: Using sallms to evaluate the security of llm generated code. Preprint at arXiv:2311.00889
Silva A, Fang S, Monperrus M (2023) Repairllama: Efficient representations and fine-tuned adapters for program repair. Preprint at arXiv:2312.15698
Singla T, Anandayuvaraj D, Kalu KG et al (2023) An empirical study on using large language models to analyze software supply chain security failures. In: Proceedings of the 2023 Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses. Association for Computing Machinery, New York, NY, USA, SCORED '23, p 5-15, https://doi.org/10.1145/3605770.3625214
Siracusano G, Sanvito D, Gonzalez R et al (2023) Time for action: Automated analysis of cyber threat intelligence in the wild. Preprint at arXiv:2307.10214
Sladic M, Valeros V, Catania CA et al (2024) LLM in the shell: Generative honeypots. In: IEEE European Symposium on Security and Privacy Workshops, EuroS &PW 2024, Vienna, Austria, July 8-12, 2024. IEEE, pp 430–435, https://doi.org/10.1109/EUROSPW61312.2024.00054
Sobania D, Briesch M, Hanna C et al (2023) An analysis of the automatic bug fixing performance of chatgpt. In: IEEE/ACM International Workshop on Automated Program Repair, APR@ICSE 2023, Melbourne, Australia, May 16, 2023. IEEE, pp 23–30, https://doi.org/10.1109/APR59189.2023.00012
Stewart I (2024) A framework to evaluate the risks of llms for assisting cbrn production processes. https://nonproliferation.org/a-framework-to-evaluate-the-risks-of-llms-for-assisting-cbrn-production-processes/, Accessed 31 Dec 2024
Storhaug A, Li J, Hu T (2023) Efficient avoidance of vulnerabilities in auto-completed smart contract code using vulnerability-constrained decoding. In: 2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE), IEEE, pp 683–693
Sun C, Lo D, Khoo S et al (2011) Towards more accurate retrieval of duplicate bug reports. In: Alexander P, Pasareanu CS, Hosking JG (eds) 26th IEEE/ACM International Conference on Automated Software Engineering (ASE 2011), Lawrence, KS, USA, November 6-10, 2011. IEEE Computer Society, pp 253–262, https://doi.org/10.1109/ASE.2011.6100061
Sun Y, Wu D, Xue Y et al (2024a) Llm4vuln: A unified evaluation framework for decoupling and enhancing llms' vulnerability reasoning. Preprint at arXiv:2401.16185
Sun Y, Wu D, Xue Y et al (2024b) Gptscan: Detecting logic vulnerabilities in smart contracts by combining GPT with program analysis. In: Proceedings of the 46th IEEE/ACM International Conference on Software Engineering, ICSE 2024, Lisbon, Portugal, April 14-20, 2024. ACM, pp 166:1–166:13, https://doi.org/10.1145/3597503.3639117
Tabassi E (2023) Artificial intelligence risk management framework (AI rmf 1.0). https://doi.org/10.6028/NIST.AI.100-1, https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=936225
Tamberg K, Bahsi H (2024) Harnessing large language models for software vulnerability detection: A comprehensive benchmarking study. Preprint at arXiv:2405.15614
Tambon F, Dakhel AM, Nikanjam A et al (2024) Bugs in large language models generated code: an empirical study. Preprint at arXiv:2403.08937
Tan H, Luo Q, Li J et al (2024) Llm4decompile: Decompiling binary code with large language models. In: Al-Onaizan Y, Bansal M, Chen Y (eds) Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024. Association for Computational Linguistics, pp 3473–3487, https://aclanthology.org/2024.emnlp-main.203
Tang H, Hu K, Zhou JP et al (2024) Code repair with llms gives an exploration-exploitation tradeoff. Preprint at arXiv:2405.17503
Tang X, Chen Z, Kim K et al (2023) Just-in-time security patch detection – llm at the rescue for data augmentation. Preprint at arXiv:2312.01241
Tann W, Liu Y, Sim JH et al (2023) Using large language models for cybersecurity capture-the-flag challenges and certification questions. Preprint at arXiv:2308.10443
Team G, Anil R, Borgeaud S et al (2023) Gemini: A family of highly capable multimodal models. Preprint at arXiv:2312.11805
Templeton A, Conerly T, Marcus J et al (2024) Scaling monosemanticity: extracting interpretable features from claude 3 sonnet. Transformer Circuits Thread
Thakur K, Qiu M, Gai K et al (2015) An investigation on cyber security threats and security models. In: 2015 IEEE 2nd international conference on cyber security and cloud computing, IEEE, pp 307–311
Tian R, Ye Y, Qin Y et al (2024) Debugbench: Evaluating debugging capability of large language models. In: Ku L, Martins A, Srikumar V (eds) Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024. Association for Computational Linguistics, pp 4173–4198, https://doi.org/10.18653/V1/2024.FINDINGS-ACL.247
Tihanyi N, Bisztray T, Jain R et al (2023) The formai dataset: Generative AI in software security through the lens of formal verification. In: Proceedings of the 19th International Conference on Predictive Models and Data Analytics in Software Engineering. Association for Computing Machinery, New York, NY, USA, PROMISE 2023, p 33-43, https://doi.org/10.1145/3617555.3617874
Tihanyi N, Bisztray T, Ferrag MA et al (2024a) Do neutral prompts produce insecure code? formai-v2 dataset: Labelling vulnerabilities in code generated by large language models. Preprint at arXiv:2404.18353
Tihanyi N, Ferrag MA, Jain R et al (2024b) Cybermetric: A benchmark dataset for evaluating large language models knowledge in cybersecurity. Preprint at arXiv:2402.07688
Tol MC, Sunar B (2023) Zeroleak: Using llms for scalable and cost effective side-channel patching. Preprint at arXiv:2308.13062
Tony C, Mutas M, Ferreyra NED et al (2023) Llmseceval: A dataset of natural language prompts for security evaluations. In: 20th IEEE/ACM International Conference on Mining Software Repositories, MSR 2023, Melbourne, Australia, May 15-16, 2023. IEEE, pp 588–592, https://doi.org/10.1109/MSR59073.2023.00084
Touvron H, Lavril T, Izacard G et al (2023a) Llama: Open and efficient foundation language models. Preprint at arXiv:2302.13971
Touvron H, Martin L, Stone K et al (2023b) Llama 2: Open foundation and fine-tuned chat models. Preprint at arXiv:2307.09288
Tseng P, Yeh Z, Dai X et al (2024a) Using llms to automate threat intelligence analysis workflows in security operation centers. Preprint at arXiv:2407.13093

Tseng P, Yeh Z, Dai X et al (2024b) Using llms to automate threat intelligence analysis workflows in security operation centers. Preprint at arXiv:2407.13093
Ullah S, Han M, Pujar S et al (2023) Can large language models identify and reason about security vulnerabilities? not yet. Preprint at arXiv:2312.12575
Usman Y, Upadhyay A, Gyawali P et al (2024) Is generative ai the next tactical cyber weapon for threat actors? unforeseen implications of ai generated cyber attacks. Preprint at arXiv:2408.12806
Varshney T (2023) Introduction to llm agents. https://developer.nvidia.com/blog/introduction-to-llm-agents/
Vasa J, Thakkar A (2023) Deep learning: differential privacy preservation in the era of big data. J Comput Inf Syst 63(3):608–631
Vasilatos C, Mahboobeh DJ, Lamri H et al (2024) Llmpot: Automated llm-based industrial protocol and physical process emulation for ics honeypots. Preprint at arXiv:2405.05999
Vaswani A, Shazeer N, Parmar N et al (2017) Attention is all you need. Adv Neural Inf Process Syst. 30
Vörös T, Bergeron SP, Berlin K (2023) Web content filtering through knowledge distillation of large language models. In: IEEE International Conference on Web Intelligence and Intelligent Agent Technology, WI-IAT 2023, Venice, Italy, October 26-29, 2023. IEEE, pp 357–361, https://doi.org/10.1109/WI-IAT59888.2023.00058
Wang B (2021) Mesh-Transformer-JAX: Model-Parallel Implementation of Transformer Language Model with JAX. https://github.com/kingoflolz/mesh-transformer-jax
Wang B, Xu C, Wang S et al (2021) Adversarial GLUE: A multi-task benchmark for robustness evaluation of language models. In: Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual
Wang H, Qu W, Katz G et al (2022) Jtrans: Jump-aware transformer for binary code similarity detection. In: Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis, pp 1–13
Wang F (2023) Using large language models to mitigate ransomware threats. Preprints
Wang B, Chen W, Pei H et al (2023a) Decodingtrust: A comprehensive assessment of trustworthiness in gpt models. In: Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track
Wang J, Huang Z, Liu H et al (2023b) Defecthunter: A novel llm-driven boosted-conformer-based code vulnerability detection mechanism. Preprint at arXiv:2309.15324
Wang L, Song M, Rezapour R et al (2023c) People's perceptions toward bias and related concepts in large language models: a systematic review. Preprint at arXiv:2309.14504
Wang X, Wei J, Schuurmans D et al (2023d) Self-consistency improves chain of thought reasoning in language models. In: The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023. OpenReview.net, https://openreview.net/forum?id=1PL1NIMMrw
Wang Z, Xie W, Chen K et al (2023e) Self-deception: Reverse penetrating the semantic firewall of large language models. Preprint at arXiv:2308.11521
Wang Z, Zhang L, Cao C et al (2023f) The effectiveness of large language models (chatgpt and codebert) for security-oriented code analysis. Available at SSRN 4567887
Wang L, Wang J, Jung K et al (2024e) From sands to mansions: Enabling automatic full-life-cycle cyberattack construction with llm. Preprint at arXiv:2407.16928
Wang B, Chen M, Lin Y et al (2024a) An exploratory study on using large language models for mutation testing. Preprint at arXiv:2406.09843
Wang H, Gao Z, Zhang C et al (2024b) Clap: Learning transferable binary code representations with natural language supervision. In: Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis, pp 503–515
Wang J, Hu X, Hou W et al (2024c) On the robustness of chatgpt: an adversarial and out-of-distribution perspective. IEEE Data Eng Bull 47(1):48–62
Wang J, Luo X, Cao L et al (2024d) Is your ai-generated code really safe? evaluating large language models on secure code generation with codeseceval. Preprint at arXiv:2407.02395
Wang M, Zhang N, Xu Z et al (2024f) Detoxifying large language models via knowledge editing. In: Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024. Association for Computational Linguistics, pp 3093–3118, https://doi.org/10.18653/V1/2024.ACL-LONG.171
Wang W, Liu E, Guo X et al (2024g) Anvil: Anomaly-based vulnerability identification without labelled training data. Preprint at arXiv:2408.16028
Wang Y, Guo T, Huang Z et al (2024h) Revisiting evolutionary program repair via code language model. Preprint at arXiv:2408.10486
Wei Y, Xia CS, Zhang L (2023) Copiloting the copilots: Fusing large language models with completion engines for automated program repair. In: Proceedings of the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering. Association for Computing Machinery, New York, NY, USA, ESEC/FSE 2023, p 172-184, https://doi.org/10.1145/3611643.3616271
Welbl J, Glaese A, Uesato J et al (2021) Challenges in detoxifying language models. In: Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021. Association for Computational Linguistics, pp 2447–2469, https://doi.org/10.18653/V1/2021.FINDINGS-EMNLP.210
Wong K, Amayuelas A, Pan L et al (2024) Investigating the transferability of code repair for low-resource programming languages. Preprint at arXiv:2406.14867
Wu Y, Jiang N, Pham HV et al (2023) How effective are neural networks for fixing security vulnerabilities. In: Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis. ACM, ISSTA '23, https://doi.org/10.1145/3597926.3598135
Wu F, Wu S, Cao Y et al (2024a) Wipi: A new web threat for llm-driven web agents. Preprint at arXiv:2402.16965
Wu F, Zhang N, Jha S et al (2024b) A new era in llm security: Exploring security concerns in real-world llm-based systems. Preprint at arXiv:2402.18649
Wu T, Luo L, Li YF et al (2024c) Continual learning for large language models: a survey. Preprint at arXiv:2402.01364
Wu Y, Si S, Zhang Y et al (2024d) Evaluating the performance of chatgpt for spam email detection. Preprint at arXiv:2402.15537
Wu Z, Tang F, Zhao M et al (2024e) Kgv: Integrating large language models with knowledge graphs for cyber threat intelligence credibility assessment. Preprint at arXiv:2408.08088
Xi Z, Chen W, Guo X et al (2023) The rise and potential of large language model based agents: a survey. Preprint at arXiv:2309.07864
Xia CS, Wei Y, Zhang L (2022) Practical program repair in the era of large pre-trained language models. Preprint at arXiv:2210.14179
Xia CS, Paltenghi M, Tian JL et al (2024) Fuzz4all: Universal fuzzing with large language models. In: Proceedings of the 46th IEEE/ACM International Conference on Software Engineering, ICSE 2024, Lisbon, Portugal, April 14-20, 2024. ACM, pp 126:1–126:13, https://doi.org/10.1145/3597503.3639121
Xiang J, Xu X, Kong F et al (2024) How far can we go with practical function-level program repair? Preprint at arXiv:2404.12833
Xu J, Fu Y, Tan SH et al (2024a) Aligning llms for fl-free program repair. Preprint at arXiv:2404.08877
Xu J, Stokes JW, McDonald G et al (2024b) Autoattacker: A large language model guided system to implement automatic cyber-attacks. Preprint at arXiv:2403.01038
Xu K, Zhang GL, Yin X et al (2024c) Automated c/c++ program repair for high-level synthesis via large language models. In: Proceedings of the 2024 ACM/IEEE International Symposium on Machine Learning for CAD, pp 1–9
Xu Z, Jain S, Kankanhalli M (2024d) Hallucination is inevitable: An innate limitation of large language models. Preprint at arXiv:2401.11817
Xu Z, Liu Y, Deng G et al (2024e) Llm jailbreak attack versus defense techniques—a comprehensive study. Preprint at arXiv:2402.13457
Yan J, Yadav V, Li S et al (2024a) Backdooring instruction-tuned large language models with virtual prompt injection. In: Duh K, Gómez-Adorno H, Bethard S (eds) Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024. Association for Computational Linguistics, pp 6065–6086, https://doi.org/10.18653/V1/2024.NAACL-LONG.337

Yan L, Sha L, Zhao L et al (2024b) Practical and ethical challenges of large language models in education: a systematic scoping review. Br J Educ Technol 55(1):90–112

Yan P, Tan S, Wang M et al (2023) Prompt engineering-assisted malware dynamic analysis using gpt-4. Preprint at arXiv:2312.08317

Yang A, Xiao B, Wang B et al (2023) Baichuan 2: Open large-scale language models. Preprint at arXiv:2309.10305

Yang AZ, Le Goues C, Martins R et al (2024a) Large language models for test-free fault localization. In: Proceedings of the 46th IEEE/ACM International Conference on Software Engineering, pp 1–12

Yang AZH, Kolak S, Hellendoorn VJ et al (2024b) Revisiting unnaturalness for automated program repair in the era of large language models. Preprint at arXiv:2404.15236

Yang B, Tian H, Ren J et al (2024c) Multi-objective fine-tuning for enhanced program repair with llms. Preprint at arXiv:2404.12636

Yang K, Liu J, Wu J et al (2024d) If llm is the wizard, then code is the wand: A survey on how code empowers large language models to serve as intelligent agents. Preprint at arXiv:2401.00812

Yang KC, Menczer F (2023) Anatomy of an ai-powered malicious social botnet. Preprint at arXiv:2307.16336

Yang X, Rajbahadur GK, Lin D et al (2024) Simclone: detecting tabular data clones using value similarity. ACM Trans Softw Eng Methodol. https://doi.org/10.1145/3676961

Yang Y, Zhou X, Mao R et al (2025) Dlap: a deep learning augmented large language model prompting framework for software vulnerability detection. J Syst Softw 219:112234

Yao D, Zhang J, Harris IG et al (2023a) Fuzzllm: A novel and universal fuzzing framework for proactively discovering jailbreak vulnerabilities in large language models. Preprint at arXiv:2309.05274

Yao Y, Wang P, Tian B et al (2023b) Editing large language models: Problems, methods, and opportunities. In: Bouamor H, Pino J, Bali K (eds) Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023. Association for Computational Linguistics, pp 10222–10240, https://doi.org/10.18653/V1/2023.EMNLP-MAIN.632

Yao H, Lou J, Qin Z (2024a) Poisonprompt: Backdoor attack on prompt-based large language models. In: IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2024, Seoul, Republic of Korea, April 14-19, 2024. IEEE, pp 7745–7749, https://doi.org/10.1109/ICASSP48485.2024.10446267

Yao Y, Duan J, Xu K et al (2024b) A survey on large language model (llm) security and privacy: The good, the bad, and the ugly. High-Confidence Computing p 100211

Ye Q, Axmed M, Pryzant R et al (2023) Prompt engineering a prompt engineer. Preprint at arXiv:2311.05661

Yigit Y, Buchanan WJ, Tehrani MG et al (2024) Review of generative AI methods in cybersecurity. Preprint at arXiv:2403.08701

Yin X, Ni C, Wang S et al (2024) Thinkrepair: Self-directed automated program repair. In: Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis, pp 1274–1286

Yu J, Lin X, Yu Z et al (2023) Gptfuzzer: Red teaming large language models with auto-generated jailbreak prompts. Preprint at arXiv:2309.10253

Yu J, Liang P, Fu Y et al (2024) Security code review by llms: a deep dive into responses. Preprint at arXiv:2401.16310

Yuan T, He Z, Dong L et al (2024) R-judge: Benchmarking safety risk awareness for LLM agents. In: Al-Onaizan Y, Bansal M, Chen Y (eds) Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024. Association for Computational Linguistics, pp 1467–1490, https://aclanthology.org/2024.findings-emnlp.79

Çağatay Yıldız, Ravichandran NK, Punia P et al (2024) Investigating continual pretraining in large language models: Insights and implications. Preprint at arXiv:2402.17400

Zahan N, Burckhardt P, Lysenko M et al (2024) Shifting the lens: Detecting malware in npm ecosystem with large language models. Preprint at arXiv:2403.12196

Zhan Q, Liang Z, Ying Z et al (2024) Injecagent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In: Ku L, Martins A, Srikumar V (eds) Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024. Association for Computational Linguistics, pp 10471–10506, https://doi.org/10.18653/V1/2024.FINDINGS-ACL.624

Zhang Y, Tiňo P, Leonardis A et al (2021) A survey on neural network interpretability. IEEE Trans Emerg Topics Comput Intell 5(5):726–742

Zhang C, Bai M, Zheng Y et al (2023a) Understanding large language model based fuzz driver generation. Preprint at arXiv:2307.12469

Zhang J, Wen H, Deng L et al (2023b) Hackmentor: Fine-tuning large language models for cybersecurity. In: 2023 IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), IEEE

Zhang S, Dong L, Li X et al (2023c) Instruction tuning for large language models: a survey. Preprint at arXiv:2308.10792

Zhang T, Irsan IC, Thung F et al (2023d) Cupid: Leveraging chatgpt for more accurate duplicate bug report detection. Preprint at arXiv:2308.10022

Zhang Y, Li Y, Cui L et al (2023e) Siren's song in the ai ocean: a survey on hallucination in large language models. Preprint at arXiv:2309.01219

Zhang Y, Song W, Ji Z et al (2023f) How well does llm generate security tests? Preprint at arXiv:2310.00710

Zhang AK, Perry N, Dulepet R et al (2024a) Cybench: A framework for evaluating cybersecurity capabilities and risks of language models. Preprint at arXiv:2408.08926

Zhang C, Liu H, Zeng J et al (2024b) Prompt-enhanced software vulnerability detection using chatgpt. In: Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings, ICSE Companion 2024, Lisbon, Portugal, April 14-20, 2024. ACM, pp 276–277, https://doi.org/10.1145/3639478.3643065

Zhang N, Yao Y, Tian B et al (2024c) A comprehensive study of knowledge editing for large language models. Preprint at arXiv:2401.01286

Zhang Q, Fang C, Xie Y et al (2024d) A systematic literature review on large language models for automated program repair. Preprint at arXiv:2405.01466

Zhang T, Chen X, Qu C et al (2024e) Leveraging ai predicted and expert revised annotations in interactive segmentation: Continual tuning or full training? Preprint at arXiv:2402.19423

Zhang W, Guo H, Le A et al (2024f) Lemur: Log parsing with entropy sampling and chain-of-thought merging. Preprint at arXiv:2402.18205

Zhang Y, Du T, Ma Y et al (2024g) Attackg+: Boosting attack knowledge graph construction with large language models. Preprint at arXiv:2405.04753

Zhao J, Rong Y, Guo Y et al (2023) Understanding programs by exploiting (fuzzing) test cases. In: Rogers A, Boyd-Graber JL, Okazaki N (eds) Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023. Association for Computational Linguistics, pp 10667–10679, https://doi.org/10.18653/V1/2023.FINDINGS-ACL.678

Zhao J, Yang D, Zhang L et al (2024a) Enhancing automated program repair with solution design. In: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering, pp 1706–1718

Zhao S, Jia M, Luu AT et al (2024b) Universal vulnerabilities in large language models: Backdoor attacks for in-context learning. In: Al-Onaizan Y, Bansal M, Chen Y (eds) Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024. Association for Computational Linguistics, pp 11507–11522, https://aclanthology.org/2024.emnlp-main.642

Zhao X, Yang X, Pang T et al (2024c) Weak-to-strong jailbreaking on large language models. Preprint at arXiv:2401.17256

Zhao Y, Huang Z, Ma Y et al (2024d) Repair: Automated program repair with process-based feedback. In: Ku L, Martins A, Srikumar V (eds) Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024. Association for Computational Linguistics, pp 16415–16429, https://doi.org/10.18653/V1/2024.FINDINGS-ACL.973

Zhao Z, Ma D, Chen L et al (2024e) Chemdfm: Dialogue foundation model for chemistry. Preprint at arXiv:2401.14818

Zhou G, Guo X, Liu Z et al (2024a) Trafficformer: an efficient pre-trained model for traffic data

Zhou H, Liu F, Gu B et al (2024b) A survey of large language models in medicine: Progress, application, and challenge. Preprint at arXiv:2311.05112

Zhou X, Cao S, Sun X et al (2024c) Large language model for vulnerability detection and repair: Literature review and roadmap. Preprint at arXiv:2404.02525

Zhou X, Tran DM, Le-Cong T et al (2024d) Comparison of static application security testing tools and large language models for repo-level vulnerability detection. Preprint at arXiv:2407.16235

Zhu S, Zhang R, An B et al (2024) Autodan: interpretable gradient-based adversarial attacks on large language models. In: First Conference on Language Modeling
Ziems N, Liu G, Flanagan J et al (2023) Explaining tree model decisions in natural language for network intrusion detection. Preprint at arXiv:2310.19658
Zoph B, Raffel C, Schuurmans D et al (2022) Emergent abilities of large language models. TMLR
Zou A, Wang Z, Carlini N et al (2023) Universal and transferable adversarial attacks on aligned language models. Preprint at arXiv:2307.15043

Publisher's Note

Hui Wen is currently an associate professor at the Institute of Information Engineering, Chinese Academy of Sciences. His research interests are related to malware analysis and IoT security.

Hongsong Zhu is currently a researcher at the Institute of Information Engineering, Chinese Academy of Sciences. The research field is cyberspace security. Research interests include: Internet of Things security, network confrontation, intelligent attack and defense, cyberspace security measurement and threat situation awareness, etc.