Large Language Models for Cyber Security: A Systematic Literature Review

HANXIANG XU, Huazhong University of Science and Technology, China

SHENAO WANG, Huazhong University of Science and Technology, China

NINGKE LI, Huazhong University of Science and Technology, China

KAILONG WANG*, Huazhong University of Science and Technology, China

YANJIE ZHAO, Huazhong University of Science and Technology, China

KAI CHEN*, Huazhong University of Science and Technology, China

TING YU, Hamad Bin Khalifa University, The State of Qatar

YANG LIU, Nanyang Technological University, Singapore

HAOYU WANG*, Huazhong University of Science and Technology, China

Outline

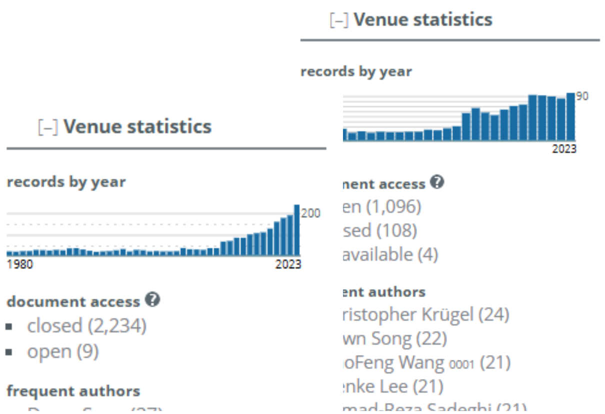
- Introduction
- What LLMs have been employed to support security tasks?
- What types of security tasks have been facilitated by LLM-based approaches?
- What domain specification technique are used to adapt LLMs to security tasks?
- What is the difference in data collection and pre-processing when applying LLMS to security tasks?
- Challenges and Opportunities

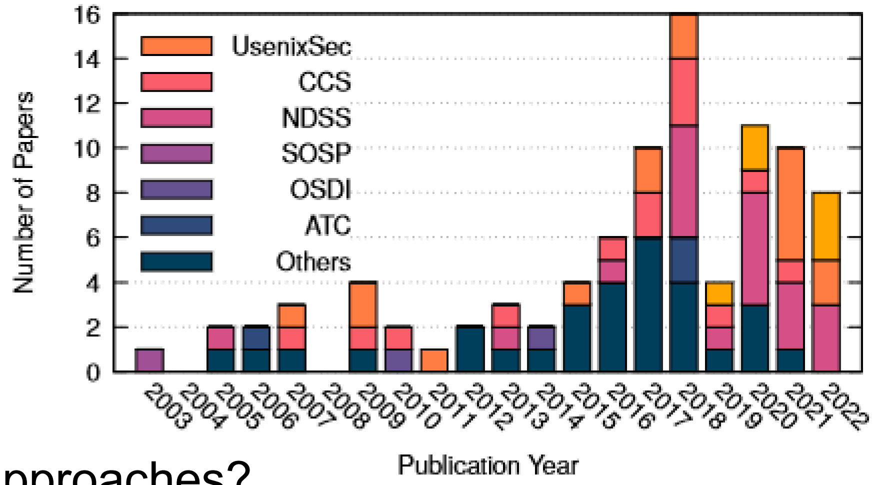
Introduction

Collecting38,112 relevant papers

This review conduct a systematic and extensive survey of the literature. By comprehensively collecting 38,112 relevant papers and systematically analyzing 127 papers from top security and software engineering venues.

Analyzing 127 papers





Research

- RQ1: What LLMs have been employed to support security tasks?
- RQ2: What types of security tasks have been facilitated by LLM-based approaches?
- RQ3: What domain specification techniques are used to adapt LLMs to security tasks?
- RQ4: What is the difference in data collection and pre-processing when applying LLMs to security tasks?

Search Strategy

Option for twelve of the top conferences and journals

Security : S&P, NDSS, USENIX, CCS, TDSC, and TIFS
Software : ICSE, ESEC/FSE, ISSTA, ASE, TOSEM, and TSE
arXiv papers of LLMs

Manual operations

ACM Digital Library, IEEE Xplore, Science Direct, Web of Science, Springer, Wiley, and arXiv.

Automated operations

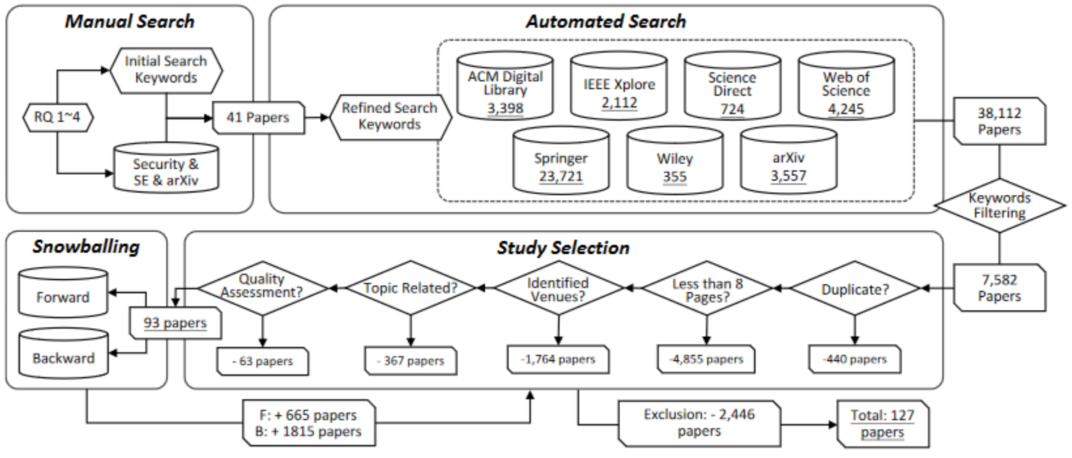


Fig. 2. Paper Search and Selection Process.

Select Strategy

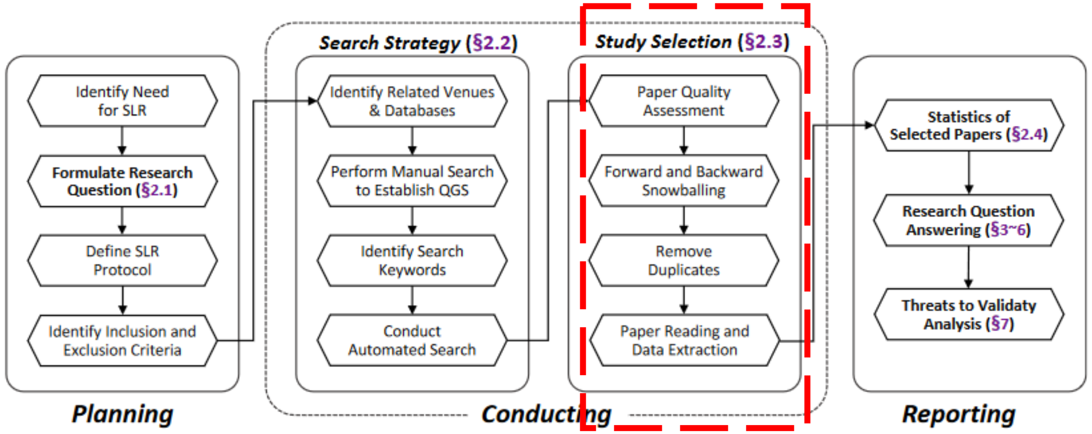
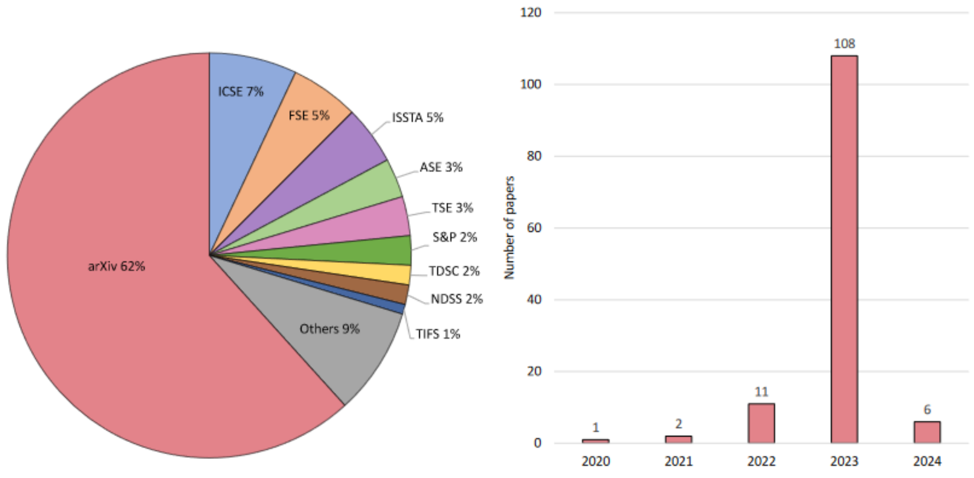


Fig. 1. Systematic Literature Review Methodology for LLM4Security.



(A) Distribution of papers across venues.
(B) Distribution of papers over years.

What LLMs have been employed to support security tasks?

Classification

- Encoder-Only

Encoder-only models comprise solely an encoder network, such as BERT and its variants, aim to predict a class label for input text .

- Encoder-Decoder

Encoder-Decoder models consist of the encoder and decoder , such as transformer , BART, T5, etc. aim to process sequence-to-sequence generation tasks

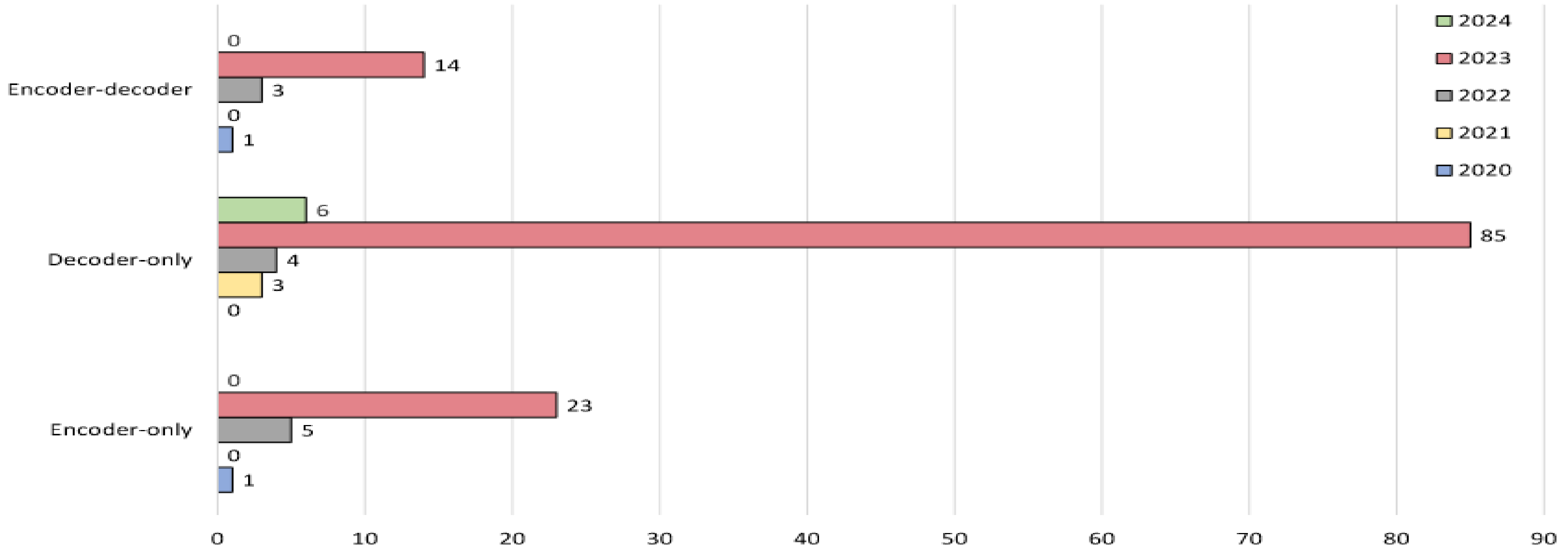
- Decoder-Only

The capability of the models to generate the target output text depends on the model's capacity to comprehend and forecast the structure, syntax, and context of language

	Model	Release Time	Open Source
Encoder-Only	BERT (8)	2018.10	Yes
	RoBERTa (12)	2019.07	Yes
	DistilBERT (3)	2019.10	Yes
	CodeBERT (8)	2020.02	Yes
	DeBERTa (1)	2020.06	Yes
	GraphCodeBERT (1)	2020.09	Yes
	CharBERT (1)	2020.11	Yes
	SecureBERT (1)	2022.04	Yes
Encoder-Decoder	T5 (4)	2019.10	Yes
	BART (1)	2019.10	Yes
	PLBART (3)	2021.03	Yes
	CodeT5 (5)	2021.09	Yes
	UniXcoder (1)	2022.03	Yes
	Flan-T5 (1)	2022.10	Yes
Decoder-Only	GPT-2 (9)	2019.02	Yes
	GPT-3 (4)	2020.04	Yes
	GPT-Neo (1)	2021.03	Yes
	CodeX (9)	2021.07	No
	CodeGen (5)	2022.03	Yes
	InCoder (1)	2022.04	Yes
	PaLM (3)	2022.04	No
	Jurassic-1 (1)	2022.04	No
	GPT-3.5 (52)	2022.11	No
	LLaMa (4)	2023.02	Yes
	GPT-4 (38)	2023.03	No
	Bard (8)	2023.03	No
	Claude (3)	2023.03	No
	StarCoder (3)	2023.05	Yes
	Falcon (2)	2023.06	Yes
	CodeLLaMa (4)	2023.08	Yes

Trend Analysis

Dominance of the decoder-only architecture in 2023 and 2024. 2023 and 2024 signaled a strong shift towards decoder-only LLMs, decoder-only LLMs have begun to take a leading role in the application of LLMs to solve security issues.



What types of security tasks have been facilitated by LLM-based approaches?

Classification

Security Domains	Security Tasks	Total
Network Security	Web fuzzing (3) Traffic and intrusion detection (10) Cyber threat analysis (5) Penetration test (4)	22
Software and System Security	Vulnerability detection (17) Vulnerability repair (10) Bug detection (8) Bug repair (20) Program fuzzing (6) Reverse engineering and binary analysis (7) Malware detection (2) System log analysis (6)	76
Information and Content Security	Phishing and scam detection (8) Harmful contents detection (6) Steganography (2) Access control (1) Forensics (1)	18
Hardware Security	Hardware vulnerability detection (2) Hardware vulnerability repair (4)	6
Blockchain Security	Smart contract security (4) Transaction anomaly detection (1)	5

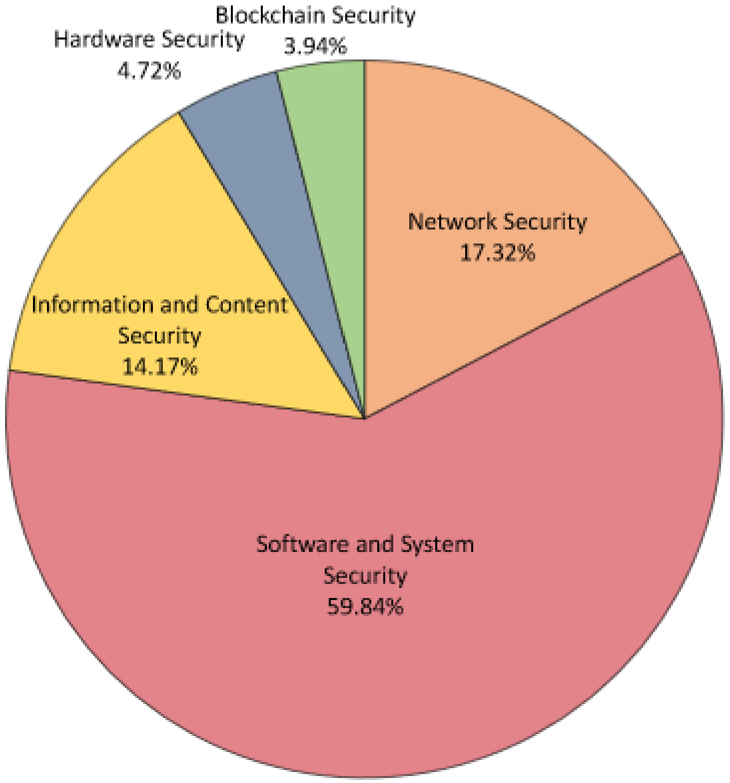


Fig. 5. Distribution of LLM usages in security domains.

Application of LLMs in Network Security

Web fuzzing & Traffic and intrusion detection

Using GPT-Fuzzer to generate fuzzy test cases.

Cyber threat analysis

LLM can generate reports for potential network security threats and predict their impact.

Penetration test

- Information gathering

Utilizing LLMs to gather information for penetration testing, including the IP address, domain information, vendor technologies, SSL/TLS credentials, and other details of the target website.

- Payload construction

LLM generates malicious payloads for penetration testing

- Vulnerability exploitation

Developing an automated Linux privilege escalation guidance tool using LLMs.

Application of LLMs in Software and System Security

Vulnerability detection & repair

Using LLM for static vulnerability detection in code

- Automated vulnerability repair tool
- Utilizing LLMs to repair side-channel vulnerabilities in programs

Bug detection & repair

LLMs can be utilized to generate code lines and compare them with the original code to flag potential bugs within code snippets .

Utilizing LLMs produces patches for various types of errors and defects

Program fuzzing

LLM Generating Fuzzy Program Test Cases

Malware detection & System log analysis

Building malware variants

Utilizing the language understanding capabilities of LLMs to identify and analyze anomalies in log data.

Application of LLMs in Information and Content Security

Phishing and scam detection

- LLMs produce phishing emails
- LLMs detect phishing emails in spam emails

Access control

PassGPT, a password generation model leveraging LLMs, introduces guided password generation, wherein PassGPT’s sampling process generates passwords adhering to user-defined constraints.

Harmful contents detection

- Detection of extreme political stances
- Tracking of criminal activity discourse identification of social media bots

Forensics

- File identification
- Evidence retrieval
- Incident response

What domain specification technique are used to adapt LLMs to security tasks?

Fine-tuning LLMs for Security Tasks

Full fine-tuning

Full fine-tuning involves adjusting all parameters of the LLMs, including every layer of the model.

Partial fine-tuning

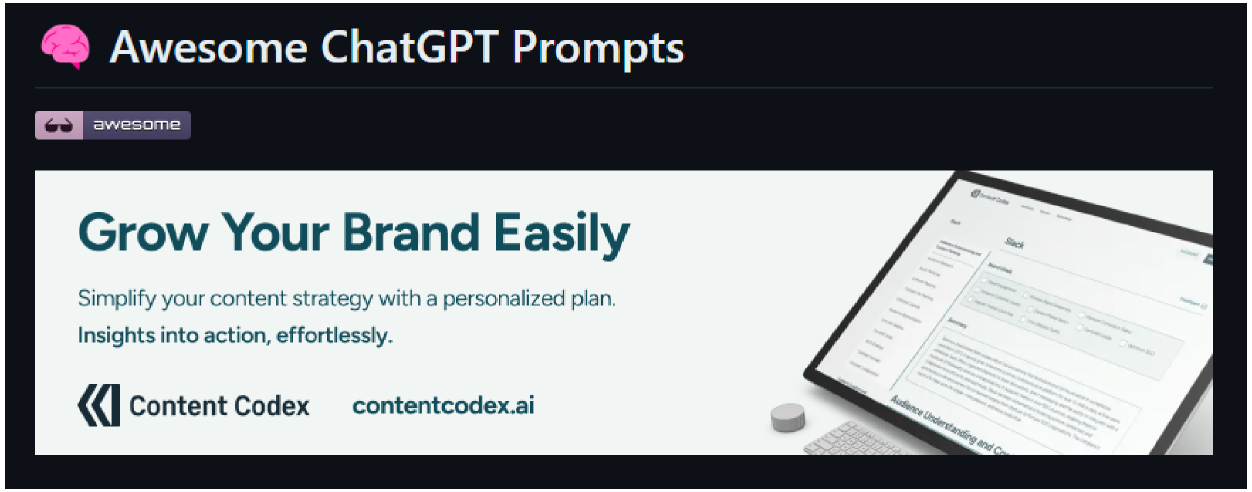
Partial fine-tuning involves updating only the top layers or a few layers of the model during the fine-tuning process.

The most popular way

Fine-tuning technique	Security task	Reference
Full fine-tuning	Bug detection (1)	[56]
	Access control (1)	
	Steganography (1)	[206]
	Reverse engineering and binary analysis (1)	[192]
	Traffic and intrusion detection (1)	[61]
	Phishing and scam detection (2)	[171] [89]
	Harmful contents detection (2)	[72] [134]
	System log analysis (2)	[96] [71]
	Vulnerability repair (3)	[217] [64] [239]
	Bug repair (4)	[233] [156] [87] [208]
	Vulnerability detection (5)	[37] [60] [198] [245] [97]
Partial fine-tuning	Traffic and intrusion detection (1)	[9]
	Harmful contens detection (1)	[82]
	Program fuzzing (1)	[46]
	Bug repair (2)	[91] [187]
	Bug detection (2)	[105] [226]
	Vulnerability detection (2)	[35] [97]

Prompting LLMs for Security Tasks

These prompts direct the large language models towards generating specific outputs while also serving as an interface for tapping into the vast knowledge encapsulated within these models.



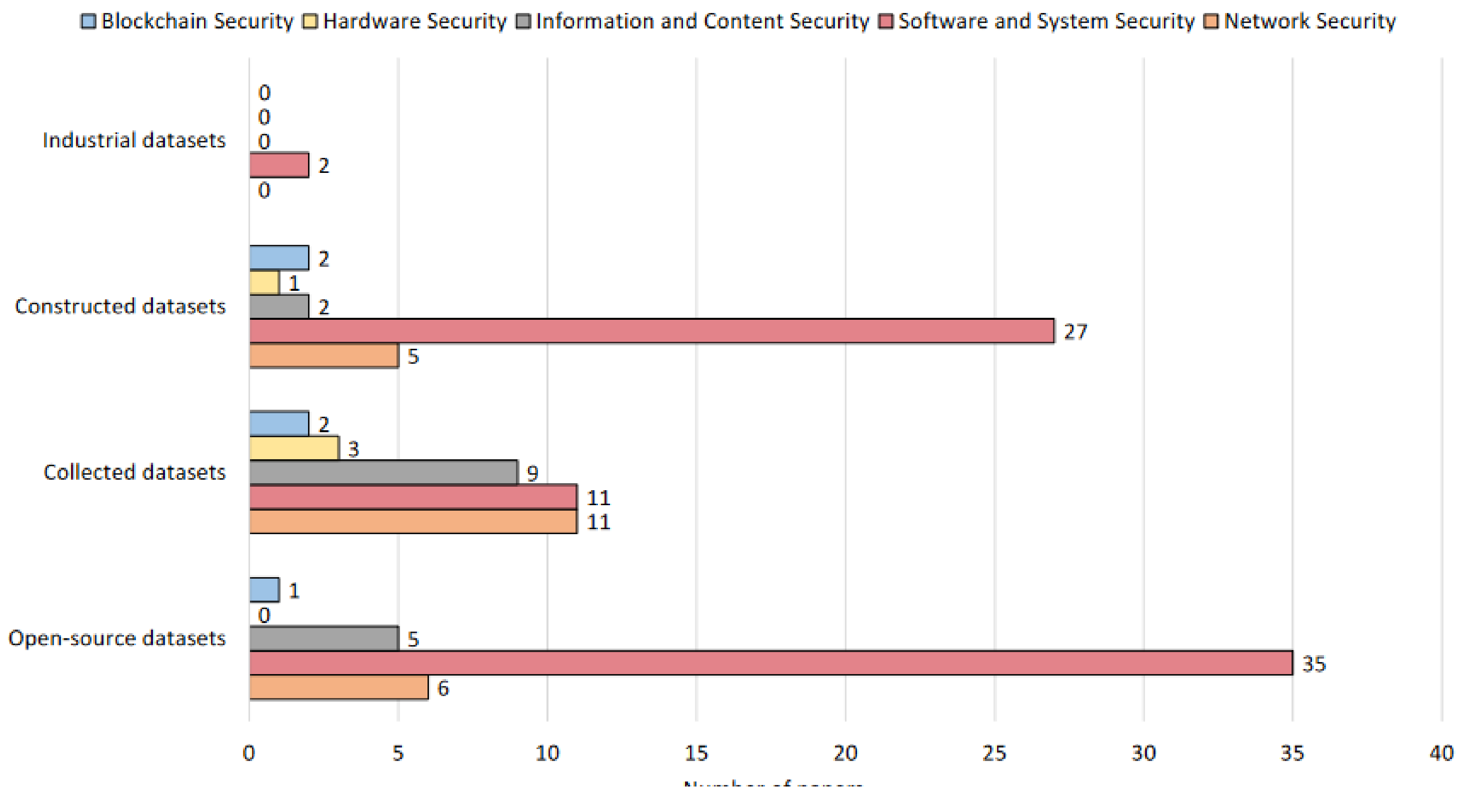
External Augmentation

These external augmentation techniques facilitate improved interaction with LLMs, bridging gaps in their knowledge base, and maximizing their capability to produce dependable outputs based on their existing knowledge.

Augmentation technique	Description	Examples	Reference
Features augmentation	Incorporating task-relevant features implicitly present in the dataset into prompts.	Adding bug descriptions, bug locations, code context or resampling for imbalanced traffic.	[91] [236] [219] [89] [9] [241] [120]
External retrieval	Retrieving task-relevant information available in external knowledge bases as input.	An external structured corpus of network threat intelligence,a hybrid patch retriever for fix pattern mining.	[53] [208] [161]
External tools	Analysis results from specialized tools serving as auxiliary inputs.	Static code analysis tools, penetration testing tools.	[73] [11] [14]
Task-adaptive training	Different training strategies from pre-training to enhance the model's adaptability to the task.	Contrastive learning, transfer learning, reinforcement learning, distillation.	[56] [105] [239] [159] [71] [114] [196] [26]
Inter-model interaction	Introducing multiple models (which can be LLMs or other models) to collaborate and interact.	Multiple LLMs feedback collaboration, graph neural networks	[26] [227] [196]
Rebroadcasting	Applicable to multi-step tasks, broad-casting the output results of each step iteratively as part of the prompt for the next step.	Difficulty-based patch example replay, variables' name propagation	[225] [233]
Post-process	Customizing special processing strategies for LLMs' outputs to better match task requirements.	Post-processing based on Levenshtein distance to mitigate hallucinations, formal verification for generated code	[35] [199]

What is the difference in data collection and pre-processing when applying LLMS to security tasks?

Data Collection



Types of Datasets

Table 8. Data types of datasets involved in prior studies.

Category	Data type	Studies	Total	References
Code-based datasets	Vulnerable code	17	71	[37] [60] [35] [39] [123] [198] [237] [245] [97] [202] [120] [217] [29] [11] [6] [31] [203]
	Source code	15		[16] [84] [45] [227] [225] [192] [14] [159] [190] [120] [193] [41] [86] [65] [85]
	Bug-fix pairs	14		[91] [113] [243] [233] [156] [146] [221] [87] [240] [223] [222] [208] [241] [187]
	Bugs	7		[110] [105] [189] [156] [87] [46] [7]
	Traffic packages	4		[137] [61] [130] [10]
	Patches	3		[97] [104] [196]
	Code changes	3		[226] [53] [213]
	Vulnerability-fix pairs	2		[239] [64]
	Bug fixing commits	2		[243] [208]
	Web attack payloads	2		[119] [114]
	Subject protocol programs	1		[132]
	Vulnerable programs	1		[157]
Text-based datasets	Prompts	17	49	[9] [139] [44] [197] [30] [73] [55] [200] [236] [158] [22] [76] [175] [131] [181] [178] [115]
	Log messages	6		[118] [38] [96] [165] [71] [184]
	Social media contents	5		[72] [82] [26] [134] [206]
	Spam messages	4		[101] [138] [27] [89]
	Bug reports	3		[56] [105] [53]
	Attack descriptions	2		[23] [57]
	CVE reports	2		[2] [3]
	Cyber threat intelligence data	2		[171] [99]
	Top-level domains	1		[122]
	Security reports	1		[4]
	Threat reports	1		[188]
	Structured threat information	1		[161]
	Program documentations	1		[219]
	Antivirus scan reports	1		[92]
	Passwords	1		[172]
	Hardware documentations	1		[133]
Combined datasets	Vulnerable code and vulnerability descriptions	2	2	[123] [35]

Data Pre-processing

Code-Based

- Data extraction
- Duplicated instance deletion
- Unqualified data deletion
- Code representation
- Data segmentation

Preprocessing techniques	Description	Examples	References
Data extraction	Retrieve appropriate text from documentation based on various software engineering tasks.	Attack description, bug reports, social media content, hardware documentation, etc.	[57] [2] [219] [72] [133]
Initial data segmentation	Categorize data into distinct groups as needed.	Split data into sentences or words.	[101] [134] [3]
Unqualified data deletion	Delete invalid text data according to the specified rules.	Remove certain symbols and words (rare words, stop words, etc.), or convert all content to lowercase.	[56] [26] [4]
Text representation	Token-based text representation.	Tokenize the texts, sentences, or words into tokens.	[3] [133]
Data segmentation	Divide the dataset into training, validation, and testing subsets for model training, parameter tuning, and performance evaluation.	Partition the dataset based on specific criteria, which may include division into training, validation, or testing subsets.	[172] [206] [122]

Preprocessing techniques	Description	Examples	References
Data extraction	Retrieve pertinent code segments from code-based datasets tailored to specific security tasks, accommodating various levels of granularity and specific task demands.	Token-level, statement-level, class-level, traffic flow.	[192] [28] [137]
Duplicated instance deletion	Eliminate duplicate instances from the dataset to maintain data integrity and avoid repetition during the training phase.	Removal of duplicate code, annotations, and obvious vulnerability indicators in function names.	[198] [237] [241]
Unqualified data deletion	Remove unfit data by implementing filtering criteria to preserve suitable samples, ensuring the dataset's quality and suitability for diverse security tasks.	Remove or anonymize comments and information that may provide obvious hints about the vulnerability (package, variable names, and strings,etc.).	[60] [97] [233] [156]
Code representation	Represented as tokens.	Tokenize source or binary code as tokens.	[245] [221] [87]
Data segmentation	Divide the dataset into training, validation, and testing subsets for model training, parameter tuning, and performance evaluation.	Partition the dataset based on specific criteria, which may include division into training, validation, or testing subsets.	[226] [190]

Text-Based

- Data extraction
- Duplicated instance deletion
- Unqualified data deletion
- Text representation
- Data segmentation

Challenges and Opportunities

Challenges

- Data scarcity
- Challenges in LLM Generalization Ability
- Challenges in LLM Interpretability

Opportunities

- Improvement of LLM4Security
- Enhancing LLM’s Performance in Existing Security Tasks
- Expanding LLM’s Capabilities in More Security Domains.