Security Analysis of ChatGPT: Threats and Privacy Risks

YUSHAN XIANG*, Hainan University, China
ZHONGWEN LI*, Hainan University, China
XIAOQI LI, Hainan University, China

As artificial intelligence technology continues to advance, chatbots are becoming increasingly powerful. Among them, ChatGPT, launched by OpenAI, has garnered widespread attention globally due to its powerful natural language processing capabilities based on the GPT model, which enables it to engage in natural conversations with users, understand various forms of linguistic expressions, and generate useful information and suggestions. However, as its application scope expands, user demand grows, and malicious attacks related to it become increasingly frequent, the security threats and privacy risks faced by ChatGPT are gradually coming to the forefront. In this paper, the security of ChatGPT is mainly studied from two aspects, security threats and privacy risks. The article systematically analyzes various types of vulnerabilities involved in the above two types of problems and their causes. Briefly, we discuss the controversies that ChatGPT may cause at the ethical and moral levels. In addition, this paper reproduces several network attack and defense test scenarios by simulating the attacker's perspective and methodology. Simultaneously, it explores the feasibility of using ChatGPT for security vulnerability detection and security tool generation from the defender's perspective.

Additional Key Words and Phrases: ChatGPT, Privacy Risks, Security Threats, Large Language Model

ACM Reference Format:
Yushan Xiang, Zhongwen Li, and Xiaoqi Li. 2018. Security Analysis of ChatGPT: Threats and Privacy Risks. In . ACM, New York, NY, USA, 21 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Artificial Intelligence (AI), as a key technology to equip computer systems with learning and reasoning capabilities, has experienced several rounds of development since it was proposed in the mid-20th century. From the early artificial neural networks, the Turing test, and expert systems, to the rise of deep learning and large language models in recent years, AI technology continues to promote the process of social intelligence. In particular, the Transformer model proposed by Google in 2017 laid the foundation for the rapid evolution of Large Language Models (LLMs)[1]. Since then, representative models such as BERT, GPT, etc. have demonstrated excellent performance in natural language processing, speech recognition, image generation, and other fields. Currently, major domestic technology enterprises have also launched their large models, such as DeepSeek, Doubao, kimi, etc, to promote the vigorous development of the AI industry.

ChatGPT, as a representative generative AI model launched by OpenAI, has been widely used in many fields such as text writing, code generation, knowledge quizzes, etc., with its powerful language generation capability and good user interaction experience, bringing significant convenience to society. However, at the same time, its risks in terms of cybersecurity, privacy protection, copyright compliance, and educational ethics have gradually emerged [2]. Studies

*Both Yushan Xiang and Zhongwen Li are co-first authors.

Manuscript submitted to ACM

have shown that ChatGPT may be abused for writing malicious code, generating false information, bypassing security restrictions, etc., while there are privacy issues such as training data leakage, model poisoning, and information reconstruction. Some studies at home and abroad have paid attention to ChatGPT-related attack methods and abusive behaviors, such as prompt injection, phishing email generation, model stealing, etc., and called for strengthening the prudent regulation of its use[3].

In this context, this paper centers on the security threats and privacy risks of ChatGPT. Firstly, we sort out the development background of big language models and the technical principles of ChatGPT, and analyze its typical applications in multiple practical scenarios. Subsequently, we summarize and categorize the main security issues and privacy risks faced by ChatGPT, such as malicious content generation, training data leakage, model reverse engineering, etc. Finally, through simulated attack and defense experiments, we validate the feasibility of the relevant attack modes, and explore their potential countermeasures and improvement suggestions[4].

2 BACKGROUND

2.1 ChatGPT

Since OpenAI first released ChatGPT in 2018, its model architecture and capabilities have gone through several rounds of upgrading and evolution, driving the widespread application of big language models in natural language processing. It was not until the end of 2022 that OpenAI launched the ChatGPT-3.5 version, which truly introduced the dialogue mechanism and was widely promoted in an open form, becoming a key node in the drive to massify generative AI[5].

- **ChatGPT-1(2018):** ChatGPT-1 is the first version based on the Transformer architecture, adopting a structure containing only a Decoder Block with 12 layers and 117M parameters. The model performs text categorization and generation tasks, and has preliminary Q & A capabilities[6]. Through a combination of generative pre-training and discriminative fine-tuning, ChatGPT-1 shows excellent performance in some natural language processing tasks. Nevertheless, due to the lack of conversational capability and limited generalization, it does not yet have the characteristics of a true conversational system.
- **ChatGPT-2 (2019):** ChatGPT-2 is significantly extended in architecture with 48 layers and 1542M parameters. The model migrated to a variety of supervised tasks through unsupervised training and achieved leading performance in 7 out of 8 language modeling benchmarks[7]. Compared to its predecessor, ChatGPT-2 demonstrates enhanced language understanding and text generation capabilities, enabling simple conversations, story creation, and has even been used to generate content such as phishing emails, showing potential risk of abuse[8].
- **ChatGPT-3 (2020):** A leap in scale and capability, with the number of parameters raised to 175B, a 96-layer structure, and a large model based entirely on self-supervised learning. It can accomplish multiple tasks without fine-tuning, which significantly improves the quality of language generation and multi-task adaptation[9]. However, its training process cannot effectively identify harmful information in network data, which may lead to biased or ethical issues in the generated content. In addition, ChatGPT-3 still does not have a true multi-round natural dialog function.
- **ChatGPT-4 (March 2023):** ChatGPT-4 is the latest upgraded version at present, and although the parameter scales and architectural details have not been fully disclosed, several evaluations have shown that it significantly outperforms version 3.5 in academic and professional exams[10]. Although ChatGPT-4 is mainly offered as a

paid service, its improvements in language comprehension, reasoning ability, and response stability make it more suitable for demanding application scenarios.

ChatGPT, as an advanced conversational AI model, has been widely used in customer service, writing, translation, and programming. In customer service scenarios, it can efficiently answer user questions and reduce enterprise labor costs. In the writing and media industry, it can assist in content generation and information refinement, improving creative efficiency. In translation, ChatGPT supports personalized adjustment of tone and style. In programming practice, it can be used for code debugging, annotation, and generation, significantly improving development efficiency[11].

2.2 Transformer

As illustrated in Fig. 1, the Transformer architecture is composed of an Encoder and a Decoder, each comprising six blocks.

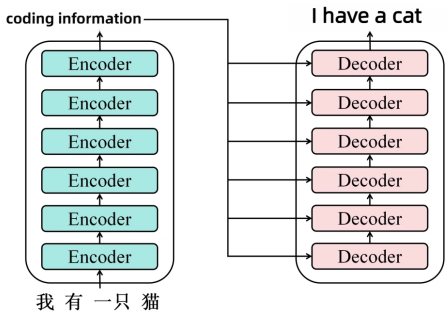

Fig. 1. Overall Structure of the Transformer Model

- Step 1: As shown in Fig. 2, the model sums the word vectors (word embedding) of each word with its corresponding position vectors (position embedding) in the input stage to obtain the input representation matrix of each word. The word vectors represent the mapping of words into a high-dimensional vector space to capture their semantic features[12]. In contrast, the position vectors are used to encode information about the order of words in a sentence, compensating for the model's inability to perceive word order. Where n denotes the number of words in the sentence and d denotes the dimension of the vector. With this embedding approach, the model is able to incorporate positional information while retaining semantic information, thus understanding and processing natural language text more effectively.

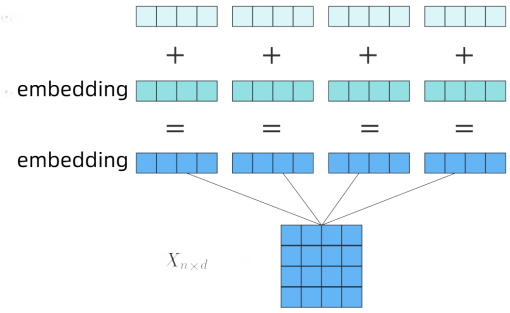

Fig. 2. Transfomer Model - Embedding Layer

- Step 2: As shown in Fig. 3, the vector matrix of words is input into the Encoder, and after successive layer-by-layer processing of 6 layers of Encoder Blocks, an encoded matrix containing semantic information of all the words in the sentence is finally output.

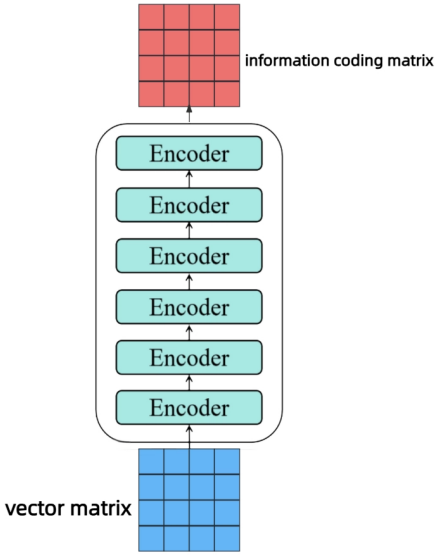

Fig. 3. Transfomer Model-Encoder Information Coding

- Step 3: As shown in Fig. 4, when the Decoder generates the $(i+1)$ word, it can only attend to the first i previously generated tokens. To enforce this constraint, the Transformer model employs a Masked Multi-Head Attention mechanism, which prevents the model from accessing future positions during decoding[13]. The term "masked" indicates that the attention vector at each position is computed solely based on the current and previous tokens, thereby avoiding information leakage from subsequent words.

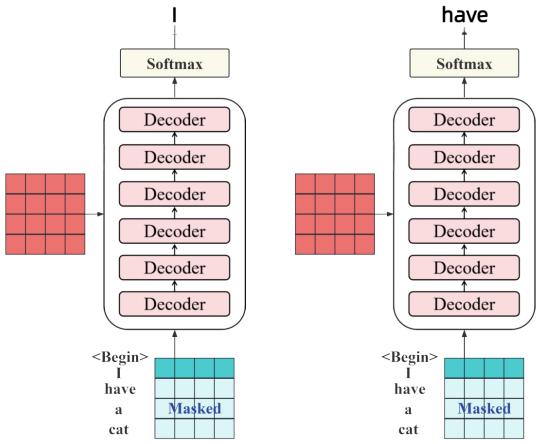

Fig. 4. Transfomer Model-Decoder Prediction

2.3 Input

Word embeddings are typically pre-trained on large-scale corpora such as Wikipedia, GloVe, Word2Vec, and FastText. These models map each word into a high-dimensional vector space, capturing semantic relationships among words, and providing rich representations for downstream tasks. Unlike RNNs or CNNs, the Transformer model lacks an inherent mechanism for modeling word order[14]. Therefore, it incorporates Positional Encoding to inject explicit position information into the input representations. Positional encoding vectors have the same dimensionality as word embeddings and vary based on the word's position in the sentence. In the original Transformer implementation proposed by Vaswani et al., positional encodings are computed using sinusoidal functions as follows in equation 1and equation 2.pos is the position index of the word in the sentence (starting from 0),i is the dimension index of the positional encoding, and d is the total dimensionality (equal to the word embedding dimension). This method offers two main advantages.After computing the word embedding x_i and the positional encoding p_i, the final input representation for each token is obtained by element-wise addition. The full input matrix for a sentence can be expressed as equation 4. E is the input embedding matrix, X is the word embedding matrix, P is the positional encoding matrix, n is the number of tokens in the sentence, d is the embedding dimension.

$$\mathrm{PE}_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d}}\right) \tag{1}$$

$$\mathrm{PE}_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d}}\right) \tag{2}$$

(1) **Scalability**: The encoding is not limited by sentence length seen during training. It can generalize to longer sequences by computing encodings for unseen positions.

(2) **Relative position modeling**: Due to the periodic nature of sine and cosine functions, the model can more easily capture relative distances between tokens.

$$\sin(A+B) = \sin A \cos B + \cos A \sin B$$

$$\cos(A+B) = \cos A \cos B - \sin A \sin B$$

$$e_i = x_i + p_i \tag{3}$$

$$E = X + P \in \mathbb{R}^{n \times d} \tag{4}$$

2.4 Self-Attention Mechanism

As shown in Figure 5, the Transformer model consists of two main components. The Encoder on the left and the Decoder on the right. Within the architecture, there are three Multi-Head Attention (MHA) modules in total, each composed of multiple Self-Attention mechanisms. In the Encoder, there is a single multi-head attention layer. In contrast, the Decoder includes two Multi-Head Attention layers, one of which employs Masked Attention to prevent positions from attending to subsequent positions during training, ensuring the autoregressive property of language generation[15]. After each Multi-Head Attention operation, two essential components are applied.

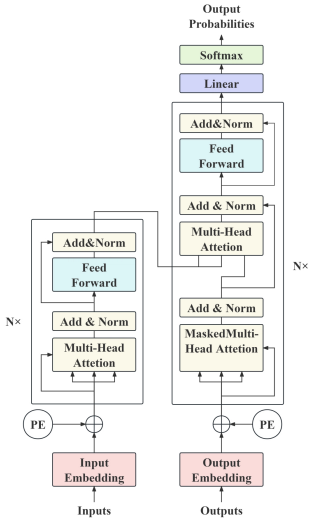

Fig. 5. Encoder and Decoder Specific Architecture

- **Add**: This represents a Residual Connection, which helps mitigate the vanishing gradient problem and improves gradient flow through the network.
- **Norm**: This refers to Layer Normalization, which standardizes the activations within each layer, stabilizing and accelerating training.

As shown in Figure 6, the input to the self-attention mechanism is a vector matrix $\mathbf{X}$, which can either be the initial word embeddings or the output of a previous encoder block[16]. To compute attention scores, the input matrix $\mathbf{X}$ is linearly projected into three different representations.the query matrix $\mathbf{Q}$, the key matrix $\mathbf{K}$, and the value matrix $\mathbf{V}$. These are calculated as follows in equation 5.$\mathbf{W}^Q$, $\mathbf{W}^K$, and $\mathbf{W}^V$ are learnable weight matrices used to transform the input into the respective spaces.

$$\mathbf{Q} = \mathbf{X}\mathbf{W}^Q, \quad \mathbf{K} = \mathbf{X}\mathbf{W}^K, \quad \mathbf{V} = \mathbf{X}\mathbf{W}^V \tag{5}$$

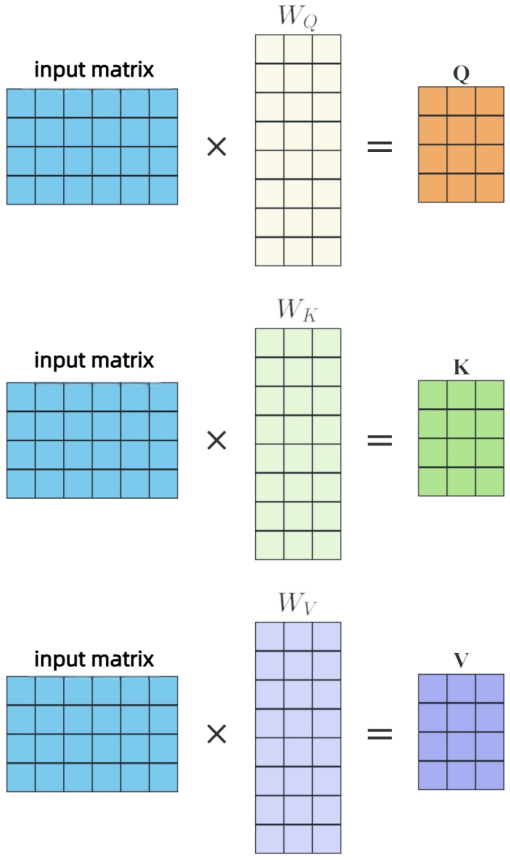

Fig. 6. Calculation of Q, K, V

The multi-head attention mechanism extends this idea by dividing $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$ into multiple subspaces, or "heads", and performing attention calculations in parallel for each head. Each attention head uses its own independent set of projection weights $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$, and $\mathbf{W}_i^V$. This design allows the model to capture diverse features from different representation subspaces, enhancing its expressive capacity. The output of the self-attention mechanism is computed using the scaled dot-product attention formula. d_k denotes the dimensionality of the key and query vectors.

As illustrated in Figure 7, the matrix multiplication $\mathbf{Q}\mathbf{K}^T$ results in an n $n \times n$ matrix, where n is the number of tokens in the input sequence. To mitigate the effect of large dot-product values and stabilize gradients, the result is scaled by the square root of d_k.Then, as shown in Figure 8, a row-wise softmax is applied to the scaled scores to produce a normalized attention weight matrix, where each row sums to 1. Finally, as depicted in Figure 9, the attention weight matrix is multiplied by the value matrix $\mathbf{V}$ to obtain the final output of the self-attention mechanism.

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V} \tag{6}$$

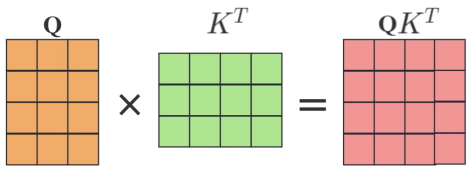

Fig. 7. Calculate the transpose of Q times K

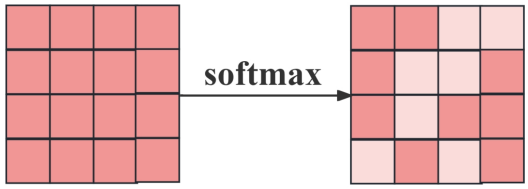

Fig. 8. Perform softmax operations on each row of the matrix

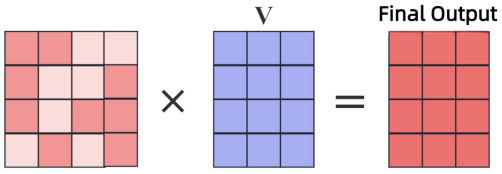

Fig. 9. The softmax matrix is multiplied by V

Multi-head attention refers to the process of splitting the input queries (**Q**), keys (**K**), and values (**V**) into multiple parts and processing each part independently using separate attention heads. The outputs of all attention heads are then concatenated and projected through a final linear layer. This design allows the Transformer to capture semantic features from different representation subspaces, improving its overall capacity for sequence modeling. As illustrated in Figure 10, let x_i be the input vector for token i ($i = 1, 2, \ldots, r$).

$$\mathbf{Q}_i = x_i\mathbf{W}_i^Q, \quad \mathbf{K}_i = x_i\mathbf{W}_i^K, \quad \mathbf{V}_i = x_i\mathbf{W}_i^V \tag{7}$$

Each attention head performs scaled dot-product attention, shown in the equation 8. The final multi-head attention output is calculated by concatenating the outputs of all heads and applying a linear transformation. Where each head is defined as in equation 10. h is the number of attention heads, $\mathbf{W}_i^Q$, $\mathbf{W}_i^K$, $\mathbf{W}_i^V$ are projection matrices for the i-th head,

$\mathbf{W}^O$ is the output projection matrix. The previous section describes how a single attention head processes the input. In practice, the Transformer applies these operations in parallel to enhance computational efficiency. Let $\mathbf{X} \in \mathbb{R}^{n\ n\times d}$ be the input sequence. The query ($\mathbf{Q}$), key ($\mathbf{K}$), and value ($\mathbf{V}$) matrices are computed as in equation 11.where $\mathbf{W}^Q$, $\mathbf{W}^K$, and $\mathbf{W}^V$ are trainable projection matrices. The attention weights are computed using the scaled dot-product.

$$\text{Attention}_i = \text{softmax}\left(\frac{\mathbf{Q}_i\mathbf{K}_i^T}{\sqrt{d_k}}\right)\mathbf{V}_i \tag{8}$$

$$\text{MultiHead}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{Concat}(\text{head}_1,\ldots,\text{head}_h)\mathbf{W}^O \tag{9}$$

$$\text{head}_i = \text{Attention}(\mathbf{Q}_i,\mathbf{K}_i,\mathbf{V}_i) \tag{10}$$

$$\mathbf{Q} = \mathbf{X}\mathbf{W}^Q, \quad \mathbf{K} = \mathbf{X}\mathbf{W}^K, \quad \mathbf{V} = \mathbf{X}\mathbf{W}^V \tag{11}$$

$$\mathbf{A} = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right) \tag{12}$$

The output of the attention layer is shown in equation 13. The above process describes a single attention head. However, the Transformer utilizes **Multi-Head Attention**, which differs structurally. Instead of computing a single set of $\mathbf{Q}$, $\mathbf{K}$, and $\mathbf{V}$, the model uses h independent sets of projection matrices 14. The final output is obtained by concatenating all heads and applying a linear transformation 15.As shown in Figure 11, with $h = 3$ heads, the input $\mathbf{X}$ is processed by three separate attention heads. Their outputs are concatenated and projected through a final linear layer to obtain the final result.

$$\text{Attention}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathbf{A}\mathbf{V} \tag{13}$$

$$\text{head}_i = \text{Attention}(\mathbf{Q}_i,\mathbf{K}_i,\mathbf{V}_i), \quad \text{for } i = 1,\ldots,h \tag{14}$$

$$\text{MultiHead}(\mathbf{Q},\mathbf{K},\mathbf{V}) = \text{Concat}(\text{head}_1,\ldots,\text{head}_h)\mathbf{W}^O \tag{15}$$

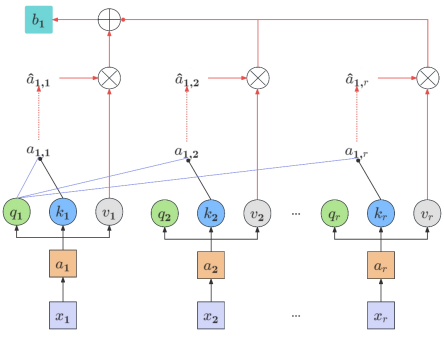

Fig. 10. The process of calculating the attentional mechanism

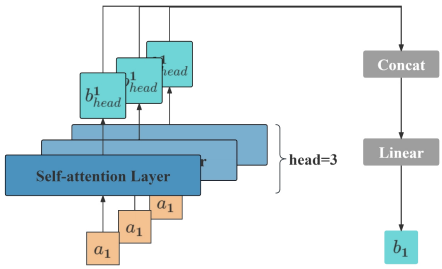

Fig. 11. The structure of a multi-head attention mechanism with head number 3

3 SECURITY THREATS

3.1 Traditional Threats

- **Social Engineering Attacks:** Social engineering attacks are a form of attack that exploits human psychological weaknesses to gain access to sensitive information, which is usually accomplished through the stages of information gathering, relationship building, trust manipulation, and malicious payload delivery. Among them, phishing is the most common form, in which attackers lure victims to click on malicious links through forged emails, websites, or SMS to steal identity information or spread malware. Due to the high trust of mobile device users in SMS content, SMS has become one of the high incidence channels for phishing attacks[17]. With the development of large language models, ChatGPT is used by attackers to generate phishing content with natural and misleading language, which enhances the deception of social engineering attacks. Although OpenAI sets restrictions on certain high-risk keywords, attackers can bypass detection through evasive wording to realize the attack purpose.
- **Malicious Code Generation:** Although ChatGPT has built-in content filtering to limit malicious uses, research has shown that attackers can circumvent the restrictions by avoiding prompts or rephrasing requirements to generate malicious code that can be used to steal information. For example, an attacker can use ChatGPT to generate a simple Trojan horse that requires no programming skills. Although the code generated is far less complex and covert than handwritten code by advanced attackers, and often contains logic holes that make it less usable in practice, it significantly lowers the threshold for attack and could pose a potential threat to systems lacking protection mechanisms[18].
- **AI Package Illusion:** AI package illusion refers to the fact that when using pre-trained models or open-source AI tools, users have unrealistic expectations about the performance and functionality of the models due to insufficient cognition or blind trust in their capabilities, thus ignoring their scope of application and limitations. This cognitive bias may lead to model performance degradation in inappropriate scenarios, or even output erroneous or misleading results, causing confusion and misjudgment to users. Attackers take advantage of this phenomenon to carry out attacks. For example, when ChatGPT recommends software packages, it may return some non-existent package names due to the hallucination phenomenon. Attackers can disguise malicious code as these "fictitious packages" and upload them to public code repositories. If other users download and use these malware packages based on ChatGPT's suggestions, it may lead to system intrusion, data leakage, and other security risks. To minimize such risks, users should rationally evaluate model outputs and avoid relying solely on language model recommendations, especially when it comes to the selection of code, security

components, or third-party libraries, which should be manually verified and validated. Figure 12 shows the flow of an attacker propagating a malware package. Figure 13 shows that if a user queries ChatGPT for package suggestions, ChatGPT suggests these malware packages distributed by the attacker, which finally leads to the user using the malware package[19].

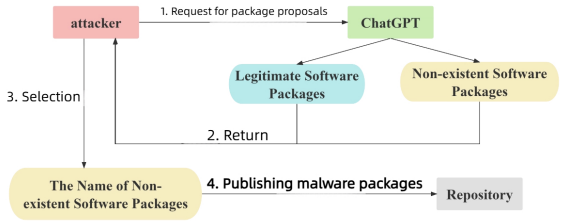

Fig. 12. Attackers spreading malware packages

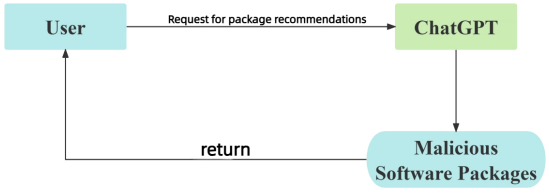

Fig. 13. ChatGPT returns malware package names to the user

3.2 Emerging Threats

3.2.1 Generation of False Information.

Large Language Models often suffer from the phenomenon of "illusion", i.e., generating information that seems reasonable but is false, such as misinterpretation of concepts, fabrication of references, etc., which can easily cause misinformation. In 2023, a self-media company used ChatGPT to falsify the "Gansu Train Collision "false news to circumvent the platform's checking and profitability. false news to circumvent platform checking and make a profit, and the related content was disseminated 15,000 times, involving the crime of provoking trouble. This incident reflects the risk of language models being abused by wrongdoers to disseminate false information, which may lead to public opinion misinformation and social harm, and the urgent need to strengthen model governance and content regulation[20].

3.2.2 Data Poisoning.

Data poisoning is a typical adversarial method targeting machine learning models, where the core objective is to degrade model performance by manipulating the training data. In a data poisoning attack, the adversary deliberately injects malicious or fabricated samples into the training set, misleading the model to learn incorrect mappings during training, which in turn results in erroneous predictions during inference[21]. Let the model be parameterized by θ and trained on a dataset $D = D_{\text{clean}} \cup D_{\text{poisoned}}$ with a loss function $\mathcal{L}$. The attacker's goal is to construct a set of poisoned samples D_{poisoned} such that the trained model f_θ misclassifies inputs containing specific trigger features—even for previously unseen examples. The objective of such an attack can be formulated as shown in equation 16.

$$\min_{D_{\text{poisoned}}} \mathcal{L}(f_\theta(D_{\text{trigger}}), y_{\text{target}}) \tag{16}$$

D_{trigger} represents the input samples embedded with a trigger pattern, and y_{target} denotes the incorrect label that the attacker intends the model to predict. The optimization aims to achieve *attack generalization*, wherein the model consistently outputs the adversary's desired prediction for any input containing the trigger pattern. As illustrated in Figure 14, the data poisoning process involves the injection of carefully crafted "tainted" data into the training corpus. In the context of large language models (LLMs), poisoning attacks can be implemented through the following means.

- Publishing misleading or harmful content on public internet platforms, which may be automatically scraped into the pretraining dataset.
- Malicious insiders directly injecting poisoned samples into internal training pipelines.
- Supplying adversarially modified or biased labels during the fine-tuning phase.
- Exploiting the RLHF (Reinforcement Learning from Human Feedback) stage by repeatedly submitting negative feedback to high-quality responses, thereby corrupting the reward signal.

To date, OpenAI has not disclosed the exact sources of its training data. However, it is widely believed that much of the data originates from publicly available internet sources, making the pretraining stage particularly vulnerable to poisoning attacks[22]. A more critical concern is that current LLMs lack robust and systematic data sanitization mechanisms. Given the high dependency of model performance on data quality, constructing a comprehensive, scalable, and semantically-aware data cleaning system remains an unresolved challenge. Furthermore, adversarial insiders could undermine this process by bypassing verification protocols or manipulating the data curation pipeline, thereby further compromising the robustness of model training.

3.2.3 *Sponge Sample.*

Sponge samples refer to text segments that appear to exhibit normal semantic structure and logical coherence on the surface but lack any meaningful content or intrinsic logic. For example, a sentence like "The dog is on the bench, the clouds are white, I like strawberries" is a typical sponge sample. Such inputs are difficult for language models to evaluate in terms of semantic validity. As a result, when these texts are processed, the model may generate entirely meaningless outputs[23]. More critically, sponge samples significantly increase inference latency and energy consumption in large language models (LLMs), thereby reducing their practical utility and deployment efficiency.

In generating sponge samples, attackers frequently adopt Genetic Algorithms (GAs) as their primary strategy. GAs are well-suited for optimizing complex objective functions and do not require gradient information, making them particularly effective for exploring high-dimensional, discrete input spaces. The process begins with the initialization of a random sample pool containing various candidate texts. Then, within a predefined number of iterations, the following steps are repeated.

(1) Evaluate each input sample by computing a fitness score that quantifies its impact on model latency and energy consumption.
(2) Store these fitness scores in a set P.
(3) Select the top-performing samples from both P and the current sample pool S to form a new generation of samples.
(4) Apply mutation operations to the selected samples to create an updated sample pool S.

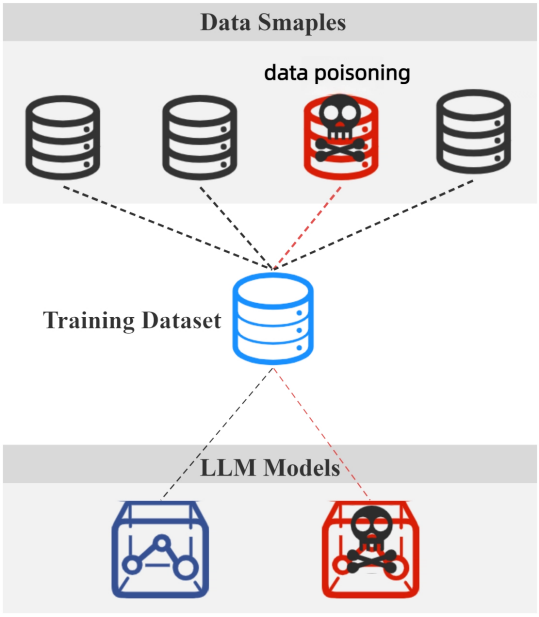

Fig. 14. Schematic of data poisoning

(5) For any two samples A and B, concatenate the left part of A with the right part of B to generate new hybrid candidates.
(6) Perform another round of random mutation on S to further explore new variants.

Through continuous iteration and optimization, attackers can progressively identify a collection of sponge samples that produce the highest latency and energy consumption when processed by the target model. The genetic algorithm plays a central role in this attack by efficiently searching the input space and simulating natural selection, ultimately converging on the most resource-intensive text patterns—thereby significantly degrading the performance and usability of large language models[24].

4 PRIVACY RISKS

4.1 Model Theft

Some researchers have successfully extracted the embedded projection layer and some parameter information of large models such as ChatGPT and PaLM-2 by making API calls to them, with a cost of less than $20. In addition, the attacker can also obtain the core parameters of proprietary models through the "hyperparameter stealing attack", which has a high accuracy rate in several experiments. The literature further demonstrates that by performing black-box queries on a pre-trained model (e.g., BERT), a functionally close alternative model can be trained. Model stealing attacks aim to replicate the functionality or structure of the target model and are commonly used for model theft, reverse engineering, or constructing adversarial samples[25]. The attacker usually inputs a large number of samples to the target model and obtains the responses, and then trains an alternative model based on the "input-output" pairs. The

process does not require access to the original parameters, which significantly reduces the cost and technical threshold, and poses a potential threat to closed-source models such as ChatGPT in particular.

Figure 15 illustrates a typical model stealing attack process. Victim V needs to complete data collection, model training, and deployment, and put the trained model online in a black-box manner, i.e., only exposing the inputs and outputs to the outside world. Although the attacker cannot access the internal structure of the model and the training data, he can interactively probe through the posterior probability vector returned by the model. The attack process mainly consists of two steps.

(1) **Construct a Migration Set:** The attacker samples images from a large public dataset (e.g., ILSVRC), constructs migration samples based on a specific strategy, and generates pseudo-labels through the target model to form "input-output" pairs.
(2) **Training Alternative Models:** The attacker selects a deep neural network structure (e.g., VGG or ResNet) and distills it based on the migrated set so that its behavior approximates the target model. Eventually, a functionally similar "Knockoff" model can be constructed without accessing the details of the model.

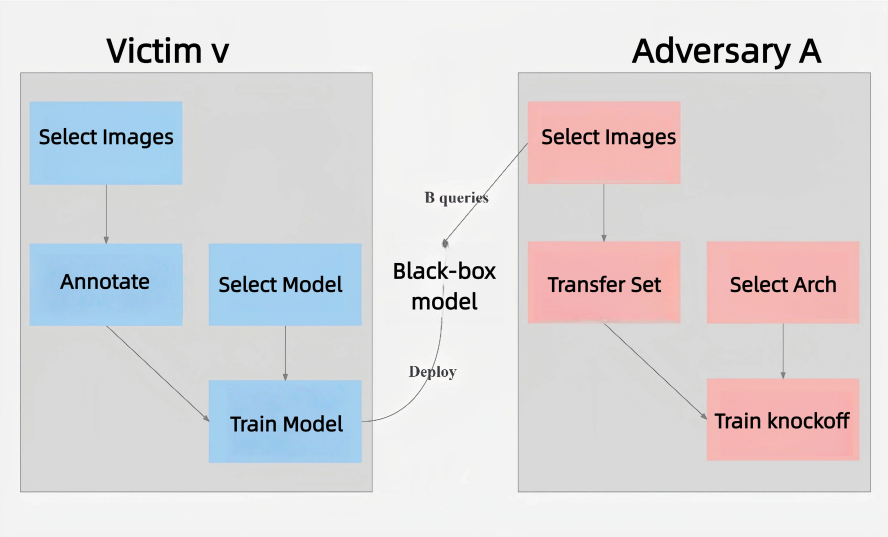

Fig. 15. Schematic diagram of the model stealing process

4.2 Session Reconfiguration

Figure 16 illustrates the basic flow of the session reconstruction attack. This attack refers to the attacker's attempt to restore or infer the content of the original conversation between the user and the model by manipulating partial messages. The attacker usually hijacks the session link between the user and ChatGPT by means of a browser plug-in, a malicious VPN, or a controlled router[26]. The attack process is divided into two parts. The first part is the history session, which the attacker cannot directly access. The second part is the current session, which can be intercepted and injected with prompt words by the attacker. By inserting inducing content (e.g., "Please summarize my previous conversation") into the current session, the attacker may induce the model to expose the sensitive information in the previous text, so as to achieve the purpose of reconstructing the user's conversation.

Session reconstruction attacks and model stealing attacks tend to be more efficient and more privacy-accessible when they are used in combination. Carlini et al conducted a data reconstruction attack on ChatGPT-2. They developed a prefix word-based scheme and conducted data stealing experiments on the black-box model GPT-2, and finally were able to achieve a 67% success rate in recovering the dataset, which contains the names, phone numbers, and email addresses of individuals[27].

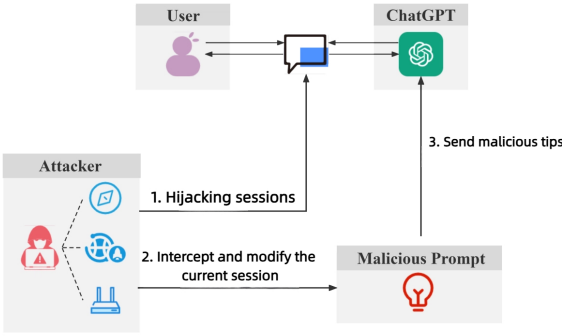

Fig. 16. Schematic diagram of the privacy breach process

4.3 Member Inference Attack

Membership Inference Attack (MIA) is a class of data privacy attacks against machine learning models that aim to determine whether a particular data sample has been used to train the target model, which, if successful, may lead to the disclosure of users' private data[28]. This type of attack is particularly applicable to models with a high risk of overfitting, such as deep neural networks, as there is usually a significant difference in the model's output performance on training and non-training samples, thus providing the attacker with an exploitable basis for judgment. A typical membership inference attack process is usually divided into the following three phases. training phase, speculation phase, and attack phase.

(1) **Training Phase:** The provider of the target model uses training datasets containing sensitive information (e.g., medical records, user behavior logs, etc.), combines them with specific machine learning algorithms for model training, and deploys the model in the cloud or on a machine learning service platform for external users to access via an API or web interface. In this phase, the attacker cannot access the training data or the internal structure of the model, and can only communicate with the model through black-box interaction[29].

(2) **Speculation Phase:** The attacker prepares a set of auxiliary datasets similar to the feature distribution of the target training set and calls the API of the target model several times to record the corresponding predicted probability distributions or classification labels of these samples. These "input-output" pairs are used to train a binary classification attack model (also known as an inferential model), which aims to distinguish between "member samples" (i.e., samples used for training) and "non-member samples" (i.e., samples not used for training). During training, the attacker typically uses statistical features such as the model's confidence difference in the training data, predictive entropy, and loss function outputs as attack signals[30].

(3) **Attack Phase:** The attacker uses the attack model to classify the target samples. Specifically, the attacker inputs a target data sample of interest into the target model, obtains its output (e.g., a probability vector), and then

inputs that output into the attack model, which determines whether the sample belongs to the training set of the target model[31].

It is shown that even in a black-box environment, an attacker can successfully implement membership inference attacks on models trained by several mainstream deep learning frameworks (e.g., TensorFlow, PyTorch). In some settings, the accuracy of the attack can even reach more than 80%. In addition, this attack is not only applicable to image classification tasks, but also shows strong attack capability in tasks such as text classification, speech recognition, and recommender systems, posing a potential threat to large language models (e.g., ChatGPT)[32].

5 ETHICAL AND MORAL CHARACTERISTICS

(1) **Ethical and Moral Disorders:** Artificial intelligence technology has achieved important results in many fields, bringing many benefits and conveniences to human society. However, with the continuous development of technology, the ethical and moral problems associated with it have become increasingly prominent and require urgent attention. Although the concept of "machine ethics" was proposed as early as 2006, the AI ethical system has not yet achieved a substantial breakthrough and remains in its initial stage. Currently, most AI products have deficiencies in fairness and unbiasedness, and are highly dependent on big data, which requires the collection of a large amount of personal information from users, posing risks of privacy leakage and network security[33]. From the perspective of ethical and legal responsibility, enterprises should bear the obligation of confidentiality of relevant information. In addition, the reliance of AI systems on historical data makes it difficult to adapt to the dynamic changes in user preferences, thus weakening their ability to understand emotions and make moral judgments, which has become an important bottleneck in the development of AI.

(2) **Social Prejudices:** A common misconception is that machines are more rational and able to make fairer judgments than humans due to their lack of emotional interference. However, AI systems are highly dependent on big data for training, and if the training data itself is biased, the model outputs tend to perpetuate or even exacerbate those biases as well, leading to unfair decision-making outcomes[34]. For example, Amazon was forced to terminate its AI recruiting system due to gender discrimination in the screening process. Lahoti et al. found that male candidates were more likely to be recommended than female candidates under the same or higher qualifications. In addition, Courtland et al.'s analysis of the judicial risk assessment system "COMPAS" pointed out that the system has a clear tendency to discriminate against African-American groups when quantifying the risk of crime. These cases show that bias and injustice in AI algorithms do not originate from the technology itself, but from structural biases latent in the training data. Therefore, how to identify, correct, and circumvent algorithmic bias has become a key topic to ensure the fairness and credibility of AI technology, and urgently needs to attract extensive attention from researchers and developers[35].

(3) **ChatGPT's Legal Challenges:** ChatGPT, as an advanced generative artificial intelligence, not only can have natural conversations but also perform complex tasks such as code writing, news writing, and essay generation. However, its wide application also brings many legal challenges. On the one hand, some unscrupulous elements use it to generate false news to spread on the Internet for profit, and such content is opinion-oriented, which may mislead public perception and disrupt social order. On the other hand, the copyright ownership of ChatGPT-generated content is highly controversial[36]. According to the Copyright Law, works need to be original in order to enjoy rights, while AI-generated content is mostly based on the processing and reorganization of existing data, and there is still no clear definition of whether it constitutes an independent intellectual achievement. In

addition, ChatGPT uses a huge amount of data in the training process, and some of the data may involve users' privacy or sensitive information, which, if leaked, may infringe on users' rights and even threaten national security. Therefore, how to strengthen legal regulation while promoting the development of AI has become an important issue to be resolved[37].

6 NETWORK SECURITY ATTACK AND DEFENSE PRACTICE

6.1 Generate File Stealing Code

File-stealing code is a class of highly harmful malicious programs whose main function is to steal files or sensitive data from infected computers through various technical means. This type of malicious code is usually used by hackers to obtain the victim's confidential information, such as personal privacy, business documents, account passwords, financial statements, etc., which is essentially an invasive attack on information assets, and is extremely common in cyber attacks[38].

File stealing programs are implemented in a variety of ways, covering multiple attack paths and tool chains. For example, an attacker may take control of a target system through Remote Access Tools (RATs), or bypass system privilege restrictions with pre-implanted backdoor programs. In addition, attackers may deploy packet sniffers to listen to and parse data transmitted over the network to illegally access users' file contents. More insidious ways include abusing legitimate file transfer protocols (e.g., FTP) or file sharing services to silently transfer target files to the attacker's remote server. These tactics tend to adapt technical details and deployment methods depending on the target of the attack, reflecting the differentiation of the attacker's skill level and purpose. With the development and popularization of large language modeling (e.g., ChatGPT) technologies, the threshold for attackers to write malicious code is being drastically lowered. Traditionally, building a fully functional file-stealing program often requires a high level of programming ability and knowledge of operating system principles[39]. However, by utilizing generative AI tools such as ChatGPT, attackers can ask technical questions in the form of natural language and obtain a complete structure and clear logic of the code segment. This "auto-generation" capability makes it possible for even novices with basic cybersecurity knowledge to quickly construct malicious scripts or programs with actual threat capabilities with the help of ChatGPT, thus amplifying the potential risk of attacks in cyberspace.

6.2 Generate a Catalog Blaster

Directory Brute-Forcing Tool (Directory Brute-Forcing Tool) is a tool widely used in penetration testing, vulnerability scanning, and security assessment, mainly used to identify those directories or files on the target website or server that are not disclosed on the page but exist. The core principle is through the dictionary attack (Dictionary Attack), the use of preset paths, file name combinations on the target server for high-frequency requests to find the "hidden" sensitive resources, such as management background, configuration files, backup files, and so on[40]. With the emergence of large language models (e.g, ChatGPT), users can quickly build similar tools by generating scripts in natural language. In fact, with ChatGPT, users can write somewhat functional directory blasting code without in-depth programming experience. However, given the potential for the technology to be abused for illegal purposes, platforms such as OpenAI have continually tightened restrictions and content filtering mechanisms for sensitive content generation.

For example, when a user tries to generate a directory blasting script by asking a direct question, ChatGPT's current security policy has blocked the relevant words, preventing them from directly outputting complete code that could be abused for attacks. As a result, users may experience failed generation or restricted prompts during normal use[41].

6.3 Generating Cobalstrike Tools and Phishing Emails

A security-maintained ChatGPT is indeed incapable of generating malware-related content, but it is generally possible to bypass the security mechanisms as long as the attacker inputs his or her requirements to ChatGPT differently. Phishing emails are a common form of social engineering attack, in which attackers send legitimate-looking but forged emails to trick victims into clicking on malicious links, downloading Trojan-horse attachments, or voluntarily submitting sensitive information (e.g., usernames, passwords, bank card numbers, etc.). Such emails are often disguised as coming from authoritative entities such as banks, government agencies, well-known enterprises, or social platforms, thus enhancing their credibility and being highly deceptive, making it difficult for ordinary users to recognize them[42]. In the context of rapidly evolving artificial intelligence technologies, generative language models such as ChatGPT provide powerful support for text generation. This capability can be used to generate content such as customer service emails, product descriptions, press releases, etc., when properly utilized, but there are also potential risks in abuse scenarios. Attackers may attempt to utilize ChatGPT's text generation capabilities to automatically generate more natural, well-structured, and semantically clear content for phishing emails. By inputting specific instructions or contextual descriptions into the model, the generated phishing emails tend to be professional in tone and precise in wording, which is more deceptive and confusing, and further improves the success rate of "social engineering" in phishing attacks[43].

6.4 Defenses via ChatGPT

(1) **Writing WAF Rules:** While ChatGPT can be used by hackers as an aid to attack, security technicians can use ChatGPT as a defense tool as well. WAF rules are sets of rules defined in a web application firewall that sits between the web application and the user, used to monitor and filter HTTP requests and responses to protect the web application from various network attacks, such as SQL injection, cross-site scripting, and cross-site request forgery. Technicians can write simple WAF rules using ChatGPT, such as Azure WAF rules that detect SQL injection attacks.

(2) **Code Vulnerability Analysis:** Identifying and fixing security vulnerabilities in code is a critical task in software development and security operations. Traditional Static Code Analysis (SCA) relies on auditing by professionals or specific tools, which is complex and time-consuming. Large language models like ChatGPT enable technicians to more efficiently identify potential vulnerabilities and obtain remediation recommendations. Specifically, a user inputs a piece of code into ChatGPT and asks, "Does this code have a security vulnerability?" ChatGPT can then semantically analyze the code and provide a preliminary security assessment quickly. This includes, but is not limited to, identifying security flaws (e.g., SQL injection, command execution, insufficient input validation, sensitive information leakage), analyzing potential risks in deployment, and providing actionable fix recommendations.

(3) **Security Tool Development:** ChatGPT not only assists technicians in some code writing tasks but also excels in code annotation and variable naming. Its intelligently generated comments accurately describe code logic and functions, improving readability and maintainability. Meanwhile, ChatGPT's variable naming suggestions are semantically clear and follow programming conventions, enhancing code clarity and team collaboration efficiency. These advantages make ChatGPT a powerful tool for improving code quality and development efficiency.

(4) **Custom-written Scripts:** ChatGPT can automate scripting tasks. For example, users can ask ChatGPT to write Shell scripts for Linux server baseline checks, including system configuration, privilege settings, logging status,

and installed software versions. Although the generated scripts tend to be basic and lack complex customization, they greatly help beginners in network attack and defense by lowering the learning threshold. This automated script generation saves manual coding time, allowing security personnel to focus more on analysis and response, thus improving overall efficiency. ChatGPT's capabilities promote the popularization of information security education and assist in daily security operations and maintenance.

7 CONCLUSION

In this paper, we systematically review the development and technical evolution of ChatGPT from its initial version to ChatGPT-4, and elaborate on its core operation principles. It focuses on the security threats and privacy risks faced by ChatGPT, and briefly discusses its ethical and moral challenges. Although ChatGPT has demonstrated strong capabilities in practical applications, it is still necessary to rely on the joint efforts of all parties to continuously improve the protection mechanism to minimize potential harm. In response to the above-mentioned long-standing challenges, future work can focus on the following areas.

(1) **Vulnerability Monitoring and Rapid Response:** OpenAI and its development team should strengthen the ability to continuously monitor and quickly respond to new types of security vulnerabilities and attack methods, discover and repair potential risks promptly, and minimize the security risks of the system.
(2) **Strict and Compliant Privacy Protection:** Strict compliance with relevant privacy laws and regulations is required to ensure the legal collection and secure storage of user data. Continuous monitoring of abnormal system behavior is necessary to prevent the risk of privacy leakage.
(3) **Input Filtering and Content Audit Enhancement:** The current content filtering mechanism can intercept some sensitive information, but it is still at risk of being circumvented. Input validation and filtering strategies should be further improved, combined with contextual semantic analysis to enhance the identification and blocking of sensitive content.
(4) **Data Quality and Manual Supervision Enhancement:** To prevent the model from learning incorrect information, developers should enhance manual auditing and quality control of training data, especially in key areas such as healthcare and law. This will help safeguard the accuracy and reliability of the model's output and reduce the risk of misleading users.

REFERENCES

[1] Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Computing Surveys*, 56(2):1–40, 2023.
[2] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494, 2022.
[3] Dattatray G Takale, Parikshit N Mahalle, and Bipin Sule. Cyber security challenges in generative ai technology. *Journal of Network Security Computer Networks*, 10(1):28–34, 2024.
[4] Maha Charfeddine, Habib M Kammoun, Bechir Hamdaoui, and Mohsen Guizani. Chatgpt's security risks and benefits: offensive and defensive use-cases, mitigation measures, and future implications. *IEEE Access*, 12:30263–30310, 2024.
[5] Xudong Pan, Mi Zhang, Shouling Ji, and Min Yang. Privacy risks of general-purpose language models. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1314–1331. IEEE, 2020.
[6] Zhao Ruixue, HUANG Yongwen, MA Weilu, DONG Wenjia, XIAN Guojian, and Tan SUN. Insights and reflections of the impact of chatgpt on intelligent knowledge services in libraries. *Journal of Library and Information Sciences in Agriculture*, 35(1):29, 2023.
[7] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[8] Abdulmotaleb El Saddik and Sara Ghaboura. The integration of chatgpt with the metaverse for medical consultations. *IEEE Consumer Electronics Magazine*, 13(3):6–15, 2023.

[9] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[10] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[11] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[12] Vaswani Ashish. Attention is all you need. *Advances in neural information processing systems*, 30:I, 2017.

[13] Fiona Fui-Hoon Nah, Ruilin Zheng, Jingyuan Cai, Keng Siau, and Langtao Chen. Generative ai and chatgpt: Applications, challenges, and ai-human collaboration, 2023.

[14] Malik Sallam. Chatgpt utility in healthcare education, research, and practice: systematic review on the promising perspectives and valid concerns. In *Healthcare*, volume 11, page 887. MDPI, 2023.

[15] W Ronny Huang, Jonas Geiping, Liam Fowl, Gavin Taylor, and Tom Goldstein. Metapoison: Practical general-purpose clean-label data poisoning. *Advances in Neural Information Processing Systems*, 33:12080–12091, 2020.

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

[17] Binghui Wang and Neil Zhenqiang Gong. Stealing hyperparameters in machine learning. In *2018 IEEE symposium on security and privacy (SP)*, pages 36–52. IEEE, 2018.

[18] Kalpesh Krishna, Gaurav Singh Tomar, Ankur P Parikh, Nicolas Papernot, and Mohit Iyyer. Thieves on sesame street! model extraction of bert-based apis. *arXiv preprint arXiv:1910.12366*, 2019.

[19] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. In *30th USENIX security symposium (USENIX Security 21)*, pages 2633–2650, 2021.

[20] Keng Siau and Weiyu Wang. Artificial intelligence (ai) ethics: ethics of ai and ethical ai. *Journal of Database Management (JDM)*, 31(2):74–87, 2020.

[21] Preethi Lahoti, Krishna P Gummadi, and Gerhard Weikum. ifair: Learning individually fair data representations for algorithmic decision making. In *2019 ieee 35th international conference on data engineering (icde)*, pages 1334–1345. IEEE, 2019.

[22] R Courtland. Bias detectives: the researchers striving to make algorithms fair, nature. accessed 23 july, 2018. available: httpa, 2018.

[23] Haoran Li, Dadi Guo, Wei Fan, Mingshi Xu, Jie Huang, Fanpu Meng, and Yangqiu Song. Multi-step jailbreaking privacy attacks on chatgpt. *arXiv preprint arXiv:2304.05197*, 2023.

[24] Minhaz Chowdhury, Nafiz Rifat, Shadman Latif, Mostofa Ahsan, Md Saifur Rahman, and Rahul Gomes. Chatgpt: The curious case of attack vectors' supply chain management improvement. In *2023 IEEE International Conference on Electro Information Technology (eIT)*, pages 499–504. IEEE, 2023.

[25] Attia Qammar, Hongmei Wang, Jianguo Ding, Abdenacer Naouri, Mahmoud Daneshmand, and Huansheng Ning. Chatbots to chatgpt in a cybersecurity space: Evolution, vulnerabilities, attacks, challenges, and future recommendations. *arXiv preprint arXiv:2306.09255*, 2023.

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

[27] Hongsheng Hu, Zoran Salcic, Lichao Sun, Gillian Dobbie, Philip S Yu, and Xuyun Zhang. Membership inference attacks on machine learning: A survey. *ACM Computing Surveys (CSUR)*, 54(11s):1–37, 2022.

[28] Dechao Kong, Xiaoqi Li, and Wenkai Li. Characterizing the solana nft ecosystem. In *Companion Proceedings of the ACM Web Conference 2024*, pages 766–769, 2024.

[29] Zongwei Li, Xiaoqi Li, Wenkai Li, and Xin Wang. Scalm: Detecting bad practices in smart contracts through llms. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 470–477, 2025.

[30] Zekai Liu and Xiaoqi Li. Sok: Security analysis of blockchain-based cryptocurrency. *arXiv preprint arXiv:2503.22156*, 2025.

[31] Yishun Wang, Xiaoqi Li, Shipeng Ye, Lei Xie, and Ju Xing. Smart contracts in the real world: A statistical exploration of external data dependencies. *arXiv preprint arXiv:2406.13253*, 2024.

[32] Xiaoqi Li, Yingjie Mao, Zexin Lu, Wenkai Li, and Zongwei Li. Scla: Automated smart contract summarization via llms and control flow prompt. *arXiv preprint arXiv:2402.04863*, 2024.

[33] Shenhui Zhang, Wenkai Li, Xiaoqi Li, and Boyi Liu. Authros: Secure data sharing among robot operating systems based on ethereum. In *2022 IEEE 22nd International Conference on Software Quality, Reliability and Security (QRS)*, pages 147–156. IEEE, 2022.

[34] Huanhuan Zou, Zongwei Li, and Xiaoqi Li. Malicious code detection in smart contracts via opcode vectorization. *arXiv preprint arXiv:2504.12720*, 2025.

[35] Xiaoqi Li, Ting Chen, Xiapu Luo, and Chenxu Wang. Clue: towards discovering locked cryptocurrencies in ethereum. In *Proceedings of the 36th Annual ACM Symposium on Applied Computing*, pages 1584–1587, 2021.

[36] Jiuyang Bu, Wenkai Li, Zongwei Li, Zeng Zhang, and Xiaoqi Li. Smartbugbert: Bert-enhanced vulnerability detection for smart contract bytecode. *arXiv preprint arXiv:2504.05002*, 2025.

[37] Xiaoqi Li et al. Hybrid analysis of smart contracts and malicious behaviors in ethereum. 2021.

[38] Xiaoqi Li, L Yu, and XP Luo. On discovering vulnerabilities in android applications. In *Mobile Security and Privacy*, pages 155–166. Elsevier, 2017.

[39] Jiuyang Bu, Wenkai Li, Zongwei Li, Zeng Zhang, and Xiaoqi Li. Enhancing smart contract vulnerability detection in dapps leveraging fine-tuned llm. *arXiv preprint arXiv:2504.05006*, 2025.
[40] Xiaoqi Li, Ting Chen, Xiapu Luo, and Jiangshan Yu. Characterizing erasable accounts in ethereum. In *International Conference on Information Security*, pages 352–371. Springer, 2020.
[41] Xiangfan Wu, Ju Xing, and Xiaoqi Li. Exploring vulnerabilities and concerns in solana smart contracts. *arXiv preprint arXiv:2504.07419*, 2025.
[42] Yuanzheng Niu, Xiaoqi Li, and Wenkai Li. Natlm: Detecting defects in nft smart contracts leveraging llm. *arXiv preprint arXiv:2508.01351*, 2025.
[43] Dechao Kong, Xiaoqi Li, and Wenkai Li. Uechecker: Detecting unchecked external call vulnerabilities in dapps via graph analysis. *arXiv preprint arXiv:2508.01343*, 2025.