Information Security Based on LLM Approaches: A Review

CHANG GONG*, School of Cyberspace Security, Hainan University, China
ZHONGWEN LI*, School of Cyberspace Security, Hainan University, China
XIAOQI LI, School of Cyberspace Security, Hainan University, China

Information security is facing increasingly severe challenges, and traditional protection means are difficult to cope with complex and changing threats. In recent years, as an emerging intelligent technology, large language models (LLMs) have shown a broad application prospect in the field of information security. In this paper, we focus on the key role of LLM in information security, systematically review its application progress in malicious behavior prediction, network threat analysis, system vulnerability detection, malicious code identification, and cryptographic algorithm optimization, and explore its potential in enhancing security protection performance. Based on neural networks and Transformer architecture, this paper analyzes the technical basis of large language models and their advantages in natural language processing tasks. It is shown that the introduction of large language modeling helps to improve the detection accuracy and reduce the false alarm rate of security systems. Finally, this paper summarizes the current application results and points out that it still faces challenges in model transparency, interpretability, and scene adaptability, among other issues. It is necessary to explore further the optimization of the model structure and the improvement of the generalization ability to realize a more intelligent and accurate information security protection system.

Additional Key Words and Phrases: Large Language Model, Information Security, Natural Language Processing, Transformer

ACM Reference Format:
Chang Gong, Zhongwen Li, and Xiaoqi Li. 2025. Information Security Based on LLM Approaches: A Review. 1, 1 (July 2025), 17 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

With the rapid development of information technology, the issue of information security has become increasingly prominent, posing a common challenge for individuals, enterprises, and even countries. Threats such as leakage of personal information, loss of financial data, and network fraud are increasing; enterprises are facing hidden dangers such as database security, system vulnerability, and intellectual property protection; and at the national level, cyberattacks and information wars are seriously jeopardizing the security of critical infrastructure and information systems [1]. Therefore, improving the security protection capability of information systems has become a crucial issue that needs to be addressed urgently. In recent years, as artificial intelligence, particularly Large Language Models (LLMs), has made significant progress in the field of natural language processing, representative models such as GPT and BERT have been widely utilized in tasks including text generation, translation, and analysis. Its powerful language understanding

*Both Chang Gong and Zhongwen Li are co-first authors of the article.

Authors' addresses: Chang Gong, School of Cyberspace Security, Hainan University, Haikou, China, 570228; Zhongwen Li, lizhongwen1230@gmail.com, School of Cyberspace Security, Hainan University, Haikou, China, 570228; Xiaoqi Li, School of Cyberspace Security, Hainan University, Haikou, China, 570228, csxqli@ieee.org.

and generation capabilities have also shown broad application prospects in the field of information security, such as malicious code recognition, network attack detection, and vulnerability analysis[2]. At the same time, the safe and effective application of large language models in the field of information security has become a new direction of study [3].

The purpose of this paper is to systematically explore the current status of the application of large language models in information security, the challenges it faces, and its practical effects. Through the combination of literature review, case study, and empirical research, we assess the advantages and shortcomings of the Large Language Model in enhancing information security capabilities and provide theoretical support and practical guidance for its further development in the field of information security.

2 BACKGROUND

2.1 Neural Network

In the field of artificial intelligence, a neural network is a mathematical model that simulates the information transfer mechanism of human neurons. Its basic structure includes an input layer, a hidden layer, and an output layer. The input layer is used to receive the original data and pass it to the subsequent network; the output layer generates the final prediction results; the hidden layer is located between the two, responsible for nonlinear mapping and feature extraction of the data, and the number of layers and nodes can be flexibly adjusted according to the task requirements; generally speaking, the deeper the hidden layer is, the stronger the model's expressive ability is. Meanwhile, the training process of neural networks mainly includes two phases: forward propagation and back propagation[4]. In the forward propagation process, the input data passes through the neurons of each layer in turn, and is processed by the weighted summation and activation function to generate the final output; in the back propagation stage, the error between the prediction result and the real label is calculated, and the algorithms such as gradient descent are used to adjust the network weights in the reverse directparticularlytimize the model performance continuously. Costing multiple iterations of training, the model error is generally reduced until convergence[5]. The structure of the neural network is shown in Fig. 1

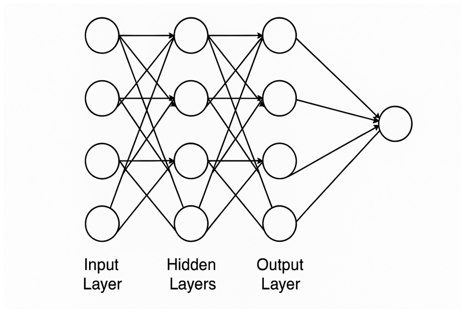

Fig. 1. Neural Network Structure Diagram

2.2 Machine Learning

- **Supervised Learning**: Supervised learning relies on labeled datasets for training, i.e., each sample contains known inputs with corresponding output targets. The model gradually optimizes the parameters and improves the prediction accuracy by continuously comparing the prediction results with the real labels. The method is widely used in scenarios such as spam recognition and credit scoring. However, the limitation of supervised learning is the high cost of acquiring labeled data, which relies on a large amount of manual labeling.

- **Unsupervised Learning**: Unsupervised learning deals with unlabeled data to discover underlying structures or patterns in the data, such as clustering and dimensionality reduction. The model automatically recognizes the intrinsic distributional features of the data without external guidance, and is commonly used for tasks such as customer segmentation and anomaly detection[6].
- **Semi-supervised Learning**: Semi-supervised learning combines a small amount of labeled data with a large amount of unlabeled data for training, taking into account the advantages of supervised and unsupervised methods[7]. In practical applications, it is particularly suitable for scenarios with high labeling costs but a large amount of data, which effectively improves the model's generalization ability and efficiency.
- **Reinforcement Learning**: Reinforcement learning is centered on the interaction of an intelligent body (agent) with the environment to maximize long-term rewards through trial-and-error mechanisms. The agent performs actions based on its current state, receives feedback (in the form of reward or punishment) from the environment, and adjusts its strategy accordingly to improve overall performance. This approach is widely used in the fields of automatic control, game strategy, and robot navigation, and is one of the important paths to realize general artificial intelligence. As shown in Fig. 2, the basic framework of reinforcement learning includes the key elements of state, action, strategy, and reward[8].
- **Deep Learning**: Deep learning is a machine learning method based on multi-layer neural networks, capable of hierarchical feature extraction and abstract representation of data. It has achieved remarkable results in the fields of image recognition, speech processing, and natural language understanding. The construction of large language models is highly dependent on deep learning, and by training on large-scale corpora, the models can capture complex linguistic structure and semantic information, and realize tasks such as syntactic analysis, context understanding, and text generation. In addition, deep learning also provides fundamental support for the application of large language models in information security. By training neural networks to identify potential threat patterns, the model can be used for security tasks such as malicious behavior detection, abnormal traffic identification, and data leakage prevention and control, showing significant potential in enhancing cyberspace governance capabilities[9].

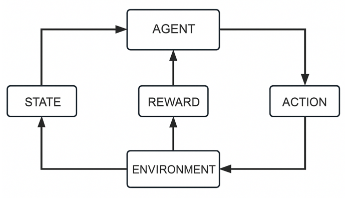

Fig. 2. Reinforcement Learning Structure Diagram

2.3 Large Model

With the continuous advancement of deep learning and computational resources, large model technology has achieved breakthroughs in the fields of natural language processing, computer vision, and speech recognition. Its development began with early artificial neural networks and back propagation algorithms, and the deep belief network proposed by Hinton in 2006 promoted the rise of deep learning, while the success of AlexNet in 2012 triggered the widespread application of deep learning. Subsequently, visual models such as VGG and ResNet have continuously optimized the network structure and improved the model performance[10]. In the field of natural language processing, pre-trained language models based on the Transformer architecture, such as BERT and GPT, have been introduced one after another, promoting a leap in language comprehension and generation capabilities. Meanwhile, systems such as AlphaGo, which incorporate multiple technologies, have also verified the potential of large models in complex intelligence tasks. Currently, the development of large models is supported by key technologies such as pre-training and fine-tuning mechanisms, distributed training, model compression, and Generative Adversarial Networks (GANs), which have become an important engine for the development of AI[11].

Large-scale pretrained language models (PLMs) are natural language understanding and generation models trained by deep learning methods using large-scale unlabeled corpora. These models usually adopt a "pre-training + fine-tuning" strategy: the pre-training phase learns linguistic features on general-purpose text, while the fine-tuning phase optimizes on labeled data according to specific tasks. The common structure is based on the Transformer framework, which consists of two main parts: the Encoder and the Decoder, which are used to model the contextual relationship between input and output sequences[12]. The current mainstream models can be roughly categorized into two types: autoregressive models (e.g., GPT), which predict the next word through the sequence antecedent, and self-encoding models (e.g., BERT), which use a masking mechanism to capture bidirectional contextual semantics. The former emphasizes text generation capabilities, while the latter is more suitable for comprehension-type tasks. Model training often combines a cross-entropy loss function with multi-task learning (e.g., MLM, NSP), designed to enhance semantic expressiveness and migration generalization. Pre-trained language models have been widely used in Q&A systems, text categorization, code generation, and information security scenarios, and have become the core supporting technology in the current field of natural language processing[13].

Language modeling based on autoregressive models: Autoregressive language models use the previous known words in the sequence as conditions to predict the next word step by step, and the whole generation process satisfies the Markov property. Typical representatives include RNNLM (Recurrent Neural Network Language Model) and Transformer-based GPT series[14]. In these models, the hidden state of each time step is determined by the information of the previous word, and the output is calculated by a Softmax layer to predict the probability. In the case of RNNLM, the training objective is to maximize the log-likelihood of each word in the entire corpus, which is expressed mathematically as follows:

$$P(w_1, w_2, \ldots, w_T) = \prod_{t=1}^{T} P(w_t \mid w_1, \ldots, w_{t-1})$$

Included among these, $P(w_t \mid w_1, \ldots, w_{t-1})$ denotes the probability of the current word under the given historical conditions. In contrast, GPT models adopt a multi-layer Transformer decoder structure and utilize a self-attention mechanism to effectively capture long-distance dependencies, thus achieving stronger context modeling capabilities. Such models perform well in tasks such as text generation and code auto-completion.

Self-Encoder Based Language Modeling: Unlike autoregression-based language models, autoencoder language models do not rely on conditional generation during training, but encode and decode the entire sentence as a whole. Typical models include Autoencoder Language Model (AELM) and BERT (Bidirectional Encoder Representations from Transformers). Taking AELM as an example, the model maps the input sequence to a low-dimensional potential space and recovers the output sequence from it. During training, the model reconstructs the original input through compression and decompression operations to minimize the reconstruction error. This error is usually measured using cross-entropy, which is defined as follows:

$$\mathcal{L}_{\text{CE}} = -\sum_{t=1}^{T}\sum_{i=1}^{V} y_{t,i} \log(\hat{y}_{t,i}) \tag{1}$$

where T denotes the length of the input sequence, V denotes the size of the vocabulary list, $y_{t,i}$ is the actual label (one-hot encoding) of the word i in the t time step, and $\hat{y}_{t,i}$ is the corresponding probability predicted by the model. The method can effectively capture the overall structural information of the input sequence and performs well in a variety of natural language processing tasks, especially for semantic understanding and context modeling[15].

The construction of a large language model usually involves the following key steps:

(1) **Data preparation:** it includes preprocessing operations such as corpus cleaning, text regularization, word splitting, vectorization, etc., to provide high-quality input data for model training.
(2) **Model construction:** The core module for constructing language models. Commonly used structures include Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and self-attention mechanisms (e.g., Transformer).
(3) **Pre-training phase:** Pre-training on a large-scale unsupervised corpus, often using strategies such as Masked Language Modeling (MLM) or autoregressive modeling, is used to learn contextual semantics by predicting masked words.
(4) **Fine-tuning phase:** Supervised fine-tuning of pre-trained models on task-specific datasets to adapt them to specific application scenarios such as classification, Q&A, and summarization. Smaller labeled datasets are usually used in this phase, and appropriate loss functions and optimization methods are selected based on the task objectives.
(5) **Integration and Testing:** Integrate and deploy the model, and optimize and adjust it using performance evaluation and error analysis to ensure its effectiveness and robustness in real tasks.

2.4 Transformer

To enhance the capability of natural language understanding and deep semantic modeling, the Transformer architecture has been proposed. Originally designed for machine translation tasks, the basic structure of the Transformer is shown in Fig. 3. The overall structure can be divided into left and right parts: the Encoder on the left side and the Decoder on the right side. In the Encoder section, each word in the input sequence is first combined with Positional Encoding through Input Embedding to generate vector representations. These representations are then sequentially processed through multiple layers consisting of Multi-Head Attention and Feed-Forward Neural Network[16], each equipped with Residual Connection and Layer Normalization (Residual Connection and Layer Normalization) between each layer. The decoder structure is similar to the encoder, but introduces Masked Multi-Head Attention after Multi-Head Attention to avoid future information leakage and receives the output from the encoder for inter-attention computation. Finally, after Linear Layer and Softmax (normalized exponential function) operations, the model outputs the corresponding translation or prediction results[17].

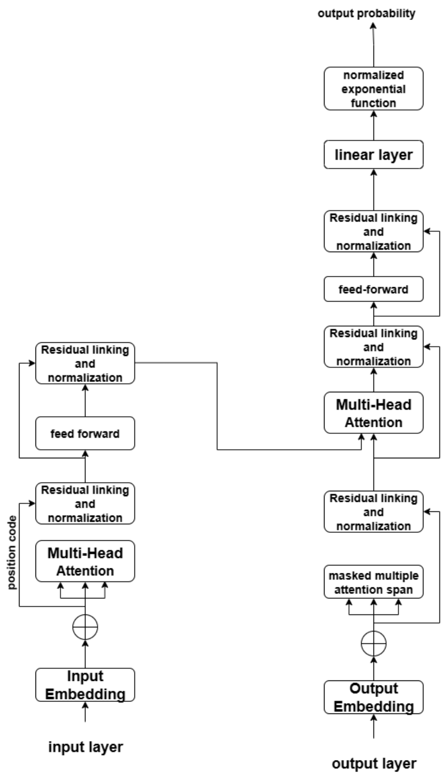
output probability
normalized exponential function
linear layer
Residual linking and normalization
feed-forward
Residual linking and normalization
Multi-Head Attention
Residual linking and normalization
masked multiple attention span
Output Embedding
output layer
Residual linking and normalization
feed forward
Residual linking and normalization
Multi-Head Attention
position code
Input Embedding
input layer

Fig. 3. Transformer Overall Architecture Diagram

2.5 Network Security Attacks

Network security attack refers to the attacker's use of technical means of computer systems, network resources, or data and information for illegal access, theft, destruction, or tampering behavior[18]. According to the different attack methods and targets, common types of network attacks mainly include:

- **Trojan Horse attack:** Embedding malicious code into a system by disguising it as a legitimate program to achieve remote control, information theft, or destructive operations without the user's knowledge.
- **Virus attack:** A type of self-replicating malicious code that spreads by infecting a host program and can cause file corruption, system crash, or information leakage.
- **Worm attack:** Self-replicating malicious code that does not depend on the host program, spreads mainly through the network, can infect quickly, and puts great pressure on network bandwidth and system resources[19].
- **Distributed Denial of Service Attack (DDoS):** The use of a large number of controlled hosts to launch concurrent requests to a target, causing it to run out of resources and disrupt its services, thereby paralyzing the system.
- **SQL Injection Attacks:** Exploit the security vulnerabilities of web applications in processing user input to control the database by injecting malicious SQL statements to realize information theft or destructive operations.

- **Social Engineering:** Attacks based on human psychological weaknesses, such as forging phishing emails, fake websites, and other means to induce users to disclose sensitive information.

In the face of evolving network attack methods, there is an urgent need to take systematic security measures[20], such as strengthening the authentication mechanism, regularly updating system patches, deploying Intrusion Detection Systems (IDS), and improving user security awareness. Meanwhile, attack detection technology also plays a key role in the network security defense system, and its core methods include:

- **Network Traffic Analysis:** Identify anomalous communication patterns and potential attack activities by analyzing the behavior of data flows at the transport and application layers.
- **Log Analysis:** Use the collection and analysis of system and application logs to achieve rapid location and response to events such as operational anomalies and intrusion behaviors.
- **Behavioral Analysis:** Modeling and anomaly detection based on user or system operational behavioral characteristics helps to discover hidden attacks such as Advanced Persistent Threats (APTs)[21].

2.6 Security Breach

A security vulnerability usually refers to a security gap in a system caused by design flaws or improper configuration in the implementation of hardware, software, or protocols, whereby an attacker can access or destroy system resources without authorization, leading to information leakage, system paralysis, or service interruption. In actual operation, vulnerabilities may trigger abnormal changes in system structure or data loss, thus bringing irreversible security risks and threatening the normal use of computer systems.

Although common security tools such as antivirus software have a certain vulnerability repair function, with the continuous evolution of attack technology, new types of vulnerabilities continue to emerge, and their hidden complexity is also increasing[22]. Some vulnerabilities are hidden in the underlying structure of the system or triggered by the interaction of multiple components, which are difficult to discover and repair promptly through traditional methods. At the same time, the continuous compression of the software development cycle has also accelerated the pace from the emergence of vulnerabilities to be exploited, so that the existing detection methods face the double challenge of efficiency and accuracy.

In addition, vulnerability detection often relies on high-level technical experts and complex detection tools, which puts high demands on resource allocation and technical capabilities. Therefore, building an efficient, intelligent, and automated vulnerability detection mechanism has become a key direction to improve network defense capability. In recent years, Large Language Models (LLMs) have shown strong potential in code semantic understanding and vulnerability prediction, promoting the transformation of vulnerability detection technology to deep learning-driven intelligence, and providing a brand new way of thinking to improve system security and robustness[23].

2.7 Malicious Code

Malicious code refers to malicious programs designed to disrupt system functionality or steal user data. Common types include viruses, worms, Trojan horses, and spyware. Viruses and worms can replicate themselves, with worms spreading autonomously over the network; Trojans use camouflage to lure users to execute them; and spyware focuses on surveillance behavior and information theft. According to the propagation path, malicious code can be spread through email, web pages, removable devices, and other media; according to the target of attack, the target can cover individual users, enterprise networks, and even government systems. Classification and characterization of malicious code are the basis for building detection and defense mechanisms. Behavioral features such as file operations, registry modifications, system calls, and network communications, as well as static attributes such as string patterns and structural features, are the important basis for malicious code identification[24].

Current mainstream detection methods include static analysis and dynamic analysis. The former analyzes the structure and logic of the source code or binary file to determine the potential risk without running the program, which is suitable for preventive detection but susceptible to obfuscation and encryption techniques; the latter monitors the program behavior during runtime, which is more reflective of the real threat but may be interfered with by the evasion mechanism of malicious code on the detection environment.

2.8 Cryptography

Cryptography is the core discipline of information security, which mainly contains two aspects of cryptographic coding and cryptanalysis. Cryptographic coding typically committed to involves generating programs to protect information, encrypting plaintext with encryption algorithms decrypting ciphertext with decryption algorithms[25]. A complete cryptographic system usually contains plaintext, ciphertext, key, encryption algorithm, and decryption algorithm, and other basic elements; the three work together to ensure the confidentiality, integrity, and availability of information.

Modern cryptosystems are divided into two categories: symmetric encryption and asymmetric encryption. Symmetric encryption uses the same key to complete the encryption and decryption process, which has the advantages of fast encryption and decryption speed and high computational efficiency, and is suitable for the protection of large amounts of data. Typical symmetric encryption algorithms include AES, DES, and so on. Asymmetric encryption is based on a pair of keys - public key (public key) and private key (private key), public key for encryption, private key for decryption. The security depends on the computational complexity of the mathematical puzzle. Common asymmetric encryption algorithms include RSA, DSA, and Elliptic Curve Cryptography (ECC)[26].

Cryptographic algorithms should be designed to satisfy the correctness requirement, i.e., decrypting the encrypted ciphertext should restore the original plaintext. In addition, according to Kirchhoff's principle, the security of a cryptosystem should depend only on the secrecy of the key; even if an attacker has all the details of the encryption algorithm, the system remains reliable as long as the key is secure. Modern cryptographic communication typically involves generating keys through key generation algorithms, encrypting plaintext with encryption algorithms, and decrypting ciphertext with decryption algorithms, thereby constituting a complete secure transmission process. The process is shown in Fig. 4.

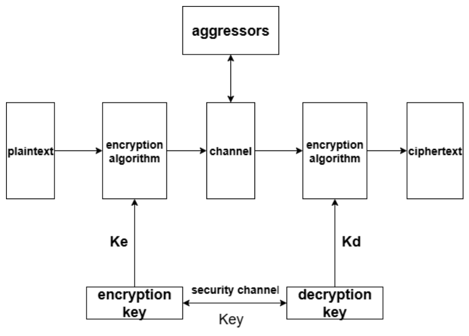

Fig. 4. Modern Cryptography

3 MALWARE DETECTION

Malware behavior identification aims to detect potential threats by analyzing the operation patterns of programs (e.g., file access, registry modification, network communication, and process creation, etc.). In recent years, large language models (LLMs) have been gradually introduced into malware detection tasks due to their excellent semantic understanding and modeling capabilities. Compared with traditional machine learning methods that rely on a large number of labeled samples, feature engineering, and computational resources, detection methods based on large language models are able to learn complex behavioral semantic patterns directly from raw data and identify unknown malicious behaviors more efficiently. By fusing pre-trained language models with behavioral sequence modeling techniques, the method shows significant advantages in improving detection accuracy and reducing false alarm rate and becomes an important means to build an intelligent malware detection system[27].

In recent years, researchers have been exploring the application of large language models and deep learning methods to malware detection and classification tasks, and have achieved remarkable results. Qiao[28] et al. propose a malware classification method that combines Word2Vec and multilayer perceptron (MLP), and the model performs well in both traditional and IoT malware classification tasks by converting byte sequences into vectors and inputting them into a deep neural network. Victor et al. experimentally verified the potential of ChatGPT for malware identification by driving a scanner that successfully detects multiple malicious processes and avoids interference from benign processes[15]. Alzaidy et al. applied RNNs and CNNs to classify Windows executable files and analyzed the impact of adversarial samples based on the results. Alzaidy et al. apply RNN and CNN to Windows executable file classification and analyze the impact of adversarial examples based on the results, which show that JSMA and C&W attacks can circumvent the model detection to a certain extent, and the robustness of the attacks varies among different models[29]. Marwaha et al. propose converting the binary files of Android malware into RGB images for visual classification using the VGG16 network. This method significantly improves classification accuracy and the F1 score, highlighting the unique advantages of visual features in malware detection[30]. The binary to malware image flow is shown in Fig. 5.

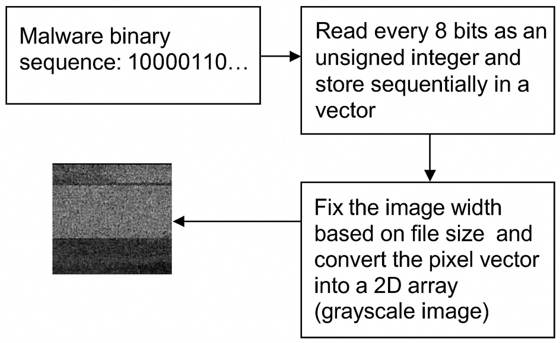
Malware binary sequence: 10000110…
Read every 8 bits as an unsigned integer and store sequentially in a vector
Fix the image width based on file size and convert the pixel vector into a 2D array (grayscale image)

Fig. 5. Binary to Malware Image

4 CYBERSECURITY ANALYSIS

4.1 DDoS Attack Prediction

As shown in Fig. 6, DDoS (Distributed Denial of Service) attacks have been continuously upgraded in recent years, presenting higher frequency, stronger attack intensity, and more complex attack methods. In response to such attacks, prior prediction of DDoS has become a key problem in the field of network security.

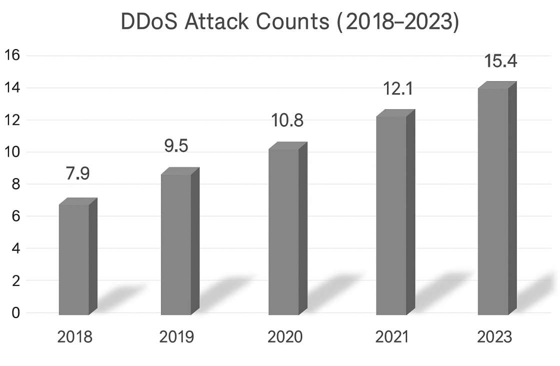

Fig. 6. Number of DDoS Attack Cases 2018-2023

With its powerful feature learning and generalization capabilities, LLM provides technical support for the efficient prediction of DDoS attacks. Through in-depth modeling of massive network traffic data, large models can automatically identify potential patterns between normal and abnormal traffic, thus upgrading the security strategy from a traditional responsive mechanism to an active prediction and defense mechanism[31]. Feature extraction is the core aspect of the large model that plays a role in DDoS attack prediction. Using deep neural networks such as Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Long Short-Term Memory Networks (LSTM), key features reflecting the state of the network, such as packet sizes, transmission intervals, communication protocols, etc., can be automatically learned from the original network stream. These models are particularly good at handling sequence data with time-dependent relationships, and thus exhibit high accuracy and robustness in anomaly detection[32].

In addition, the model training needs to rely on large-scale datasets and combine with cross-validation and test-set evaluation to ensure that the prediction models have good generalization ability and can cope with diverse DDoS attack scenarios. Ultimately, the prediction system based on the large model can analyze network traffic in real time, and quickly issue warnings and take countermeasures when anomalous patterns are identified, so as to effectively reduce the damage caused by attacks.

In practical research, Cheon et al proposed a deep learning method based on Word2Vec for DDoS attack detection[33]. The method significantly improves the model's ability to capture data patterns by encoding network traffic into semantic embeddings, which effectively enhances the detection performance. Meanwhile, Guastalla et al proposed an LLM detection strategy that combines Few-shot cue learning, Fine-tuning of the model, and neural network architecture. This strategy generates semantic understanding and fine-tuning optimization through LLM under the premise of using a small number of labeled samples, and ultimately achieves higher detection accuracy than traditional neural networks, which provides an innovative direction for DDoS defense in resource-constrained scenarios[34]. The results of DDoS attack detection in practice are as follows: Listing 1 and Tab 1.

"It's Normal Traffic" means that it is detected as normal traffic, and if it is detected as abnormal traffic, it is "Attack Traffic Detected." The port will be blocked to prevent the attack of abnormal traffic. Port to prevent the attack of abnormal traffic. The TPR metric represents the sensitivity, indicating that the model is more accurate; the F1-Score provides the reconciled mean value that represents the accuracy and sensitivity, and the experiments obtained high F1-Score values, so it can be considered valid.

```
input data [165, 164, 0.012195121951219513] prediction result ['0']
It's Normal Traffic
input data [276, 276, 0.004545454545454545] prediction result ['1']
Attack Traffic detected
Mitigation Started
attack detected from port 1
Block the port 1
attack detected from port 1
Block the port 1
```

Listing 1. Traffic Prediction Output

Table 1. Simulation Experiment Results

Metric	Value
Precision	0.915
Recall	0.612
F1-Score	0.733
Accuracy	0.747
False Positive Rate (FPR)	0.612
True Positive Rate (TPR)	0.924

4.2 Network Security Log Analysis

Network security logs are an important tool for operation and maintenance, and security teams to record the operation and event information of network devices. With the expansion of network scale and the increasing complexity of the environment, the traditional manual log analysis method is difficult to meet actual needs. In recent years, the application of Large Language Model (LLM) in security log analysis has gradually become a research hotspot. By modeling and analyzing massive logs with the help of LLM, hidden behavioral patterns and abnormal events can be effectively identified. For example, through the construction of a network log language model, abnormal access behavior can be detected, so as to warn of potential attacks timely manner. In addition, LLM can assist security teams in extracting key information from threat intelligence and automating and intellectualizing incident response.

It has been shown that deep learning and natural language processing technologies have good performance in log anomaly detection. The LogFiT model proposed by Crispin et al. is based on BERT for pre-training and fine-tuning to learn the linguistic structure of system logs[35]. The model optimizes the normal logging pattern through two objectives: mask prediction and center-of-mass distance minimization, and combines top-k prediction accuracy and center-of-mass distance in the inference stage to determine whether the log is abnormal or not. Experimental results show that LogFiT exhibits good accuracy and robustness in the log anomaly detection task.

4.3 Network Abnormal Traffic Detection

Network anomalous traffic detection is a key link in network security analysis, aiming to discover anomalous behaviors in time by monitoring traffic changes to ensure system security. In recent years, large language models (LLMs) have been gradually introduced into this field due to their powerful feature learning and pattern modeling capabilities, and show the prospect of wide application.

LLM is able to construct language models for network packets and learn the feature distribution and behavioral patterns of normal traffic. When abnormal traffic occurs, patterns that deviate from normal behavior can be identified through model comparison, achieving efficient and accurate abnormality detection. In addition, LLM can also mine the deep dependencies in the traffic to assist analysts in more accurate threat identification. The FlowTransformer model proposed by Manocchio et al[36] represents network traffic as a sequence of feature vectors (including source/destination IPs, protocol types, etc.) and uses the Transformer encoder to model its temporal dependencies to achieve classification of normal and abnormal traffic. Experiments show that this method has high accuracy and generalization ability in anomaly detection tasks. In addition, Chai et al. proposed a detection method combining CNN and Transformer, which first extracts local features by a convolutional neural network, then models global dependencies by a Transformer encoder, and finally expresses them as abstract features by a decoder, and completes the classification with the help of SVM. The method performs well in recognizing unknown attacks and potential threats, and has good detection accuracy and practical value [37].

In this paper, an anomaly detection model that fuses a convolutional neural network (CNN) with the Transformer architecture is investigated. The model mainly includes a Transformer encoder, a convolutional feature extraction layer, and a linear support vector machine (SVM) classifier, aiming to improve the accuracy of identifying anomalous behaviors of network traffic. The experimental data is selected from the Wednesday-working-hours section of the CIC-IDS-2017 intrusion detection dataset. In order to ensure the balance between training and testing, the dataset is divided into a training set and a testing set in the ratio of 80% and 20%. After the model training is completed, the first 315 samples in the test set are selected for validation, and the performance of the model is further evaluated by calculating the confusion matrix. At the same time, the model's classification effect and practical value in the anomaly detection task are comprehensively measured by combining Accuracy, Precision, Recall, and F1-score.

After training, this paper selects the first 315 samples of the test set to compute the confusion matrix and calculates the accuracy, precision, recall, and F1 score of the model. The structure is shown in Fig. 7 and Tab 2

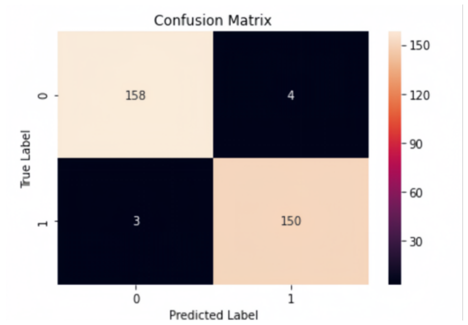

Fig. 7. Confusion Matrix

Table 2. Simulated Experiment Results

Metric	Value
Precision	0.974
Recall	0.980
F1-Score	0.977
Accuracy	0.978

5 SECURITY VULNERABILITY DETECTION

Security vulnerability detection relies on an in-depth understanding of program semantics and structure, so it is crucial to select appropriate language models. Currently, commonly used models include n-gram, RNN, LSTM, and models based on the Transformer architecture, such as BERT and GPT. The Transformer model has become a mainstream solution due to its powerful semantic modeling capability. In practice, researchers usually use large-scale pre-trained language models and fine-tune them with data from the security domain to improve the performance of the models in vulnerability detection tasks. The training process often employs a transfer learning strategy to migrate the linguistic knowledge of the general-purpose model to a specific security corpus, so as to improve the model's ability to generalize the vulnerability features.

For example, the CodeSentry framework proposed by Jones et al[38] is based on GPT-2, which encodes words and sub-words of C, C++, and Java code, and identifies the vulnerability types through the output vectors, which significantly reduces manual feature engineering.SmartConDetect, designed by Jeon et al., utilizes the BERT model to semantically analyze the Solidity smart contract. SmartConDetect by Jeon et al[39]. utilizes the BERT model to semantically analyze Solidity smart contracts and realizes automated vulnerability detection through softmax classifiers, while Szabó et al. implement static analysis in combination with the GPT API, and detect sensitive snippets of front-end applications through natural language hints to effectively identify vulnerabilities such as CWE-653[40].

The language model training method in security vulnerability detection is an important and effective technical means, making full use of deep learning and natural language processing technology, which can better discover and detect potential security vulnerabilities in the system and improve the security and stability of the system.

6 MALICIOUS CODE DETECTION AND ENCRYPTION ALGORITHM

6.1 Malicious Code Detection

The key to malicious code detection lies in the high-quality extraction and accurate classification of features. Traditional methods mainly rely on static and dynamic analysis: static analysis extracts static features from code structure and instruction sequences, which is suitable for pre-execution detection; dynamic analysis monitors system calls, network communication, and other behaviors at runtime to reveal malicious logic. Although they complement each other in practice, there are still limitations when dealing with obfuscated code, variant samples, and complex semantic structures.

The introduction of large language models (LLMs) provides a breakthrough in malicious code analysis. With their powerful capabilities in semantic modeling, context understanding, and sequence learning, LLMs can extract deeper semantic features from complex instruction sequences and API call paths, making up for the shortcomings of traditional feature engineering[41]. At the same time, the large model can generalize the identification of variant codes, which can effectively explore the latent malicious behavior patterns and significantly improve the accuracy and adaptability of detection.

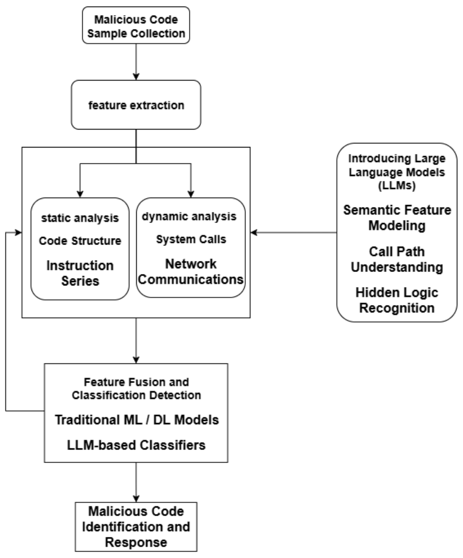
Malicious Code Sample Collection
feature extraction
static analysis
Code Structure
Instruction Series
dynamic analysis
System Calls
Network Communications
Introducing Large Language Models (LLMs)
Semantic Feature Modeling
Call Path Understanding
Hidden Logic Recognition
Feature Fusion and Classification Detection
Traditional ML / DL Models
LLM-based Classifiers
Malicious Code Identification and Response

Fig. 8. The Role of Large Language Modeling in the Malicious Code Detection Process

In the classification and detection stage, classification algorithms incorporating features extracted from LLMs have demonstrated superior performance. Combined with deep learning or supervised learning methods, researchers can build high-precision classifiers based on categories such as viruses, worms, Trojans, etc., to quickly recognize unknown malicious samples. In addition, LLMs can also take on the core modeling task in the end-to-end detection framework, which improves the robustness and practicality of the system in real attack and defense environments[42]. The Fig. 8 shows the core process of malicious code detection and highlights the key role of Large Language Models (LLMs) in feature extraction and semantic understanding. By fusing static and dynamic features, large models are able to model complex behavioral logic and improve detection accuracy.

6.2 Cryptographic Algorithms

With the continuous evolution of artificial intelligence technology, the application of large language models (LLMs) in symmetric encryption and asymmetric encryption algorithms has become a new hot spot in network security research. In symmetric encryption algorithms, LLMs can play a role in the following key aspects: first, the optimization of the key exchange mechanism, using its generation and prediction capabilities to assist in the establishment of a more secure key negotiation process; second, the improvement of encryption and decryption efficiency, through the identification of data patterns and contextual features, to improve the processing efficiency of the algorithm; third, the identification of vulnerabilities and risk assessment, the model can be used to analyze the history of the attack samples and the description of the protocol, to assist in discovering potential security risks. The model can analyze historical attack samples and protocol descriptions to assist in the discovery of potential security risks[43].

In terms of asymmetric encryption algorithms, LLMs also show application potential, mainly in key combination security assessment, digital signature logic analysis, and parameter selection optimization. With their powerful semantic understanding and reasoning capabilities, the models can be used to identify potential risks in signature structures and improve the reliability of public key infrastructure (PKI). In addition, LLMs also show some breakthrough potential in cryptography algorithm cracking research. Compared with traditional brute-force cracking and mathematical derivation, LLMs can predict potential encryption structures by learning a large number of historical key patterns, cryptographic phrase construction laws, and user behavioral data, thus assisting in reducing the difficulty of cracking. In practical applications, this method is expected to improve efficiency in specific weak password detection and model-assisted key reconstruction scenarios. The architecture of the large Language Model for modern cryptographic systems is shown in Fig. 9

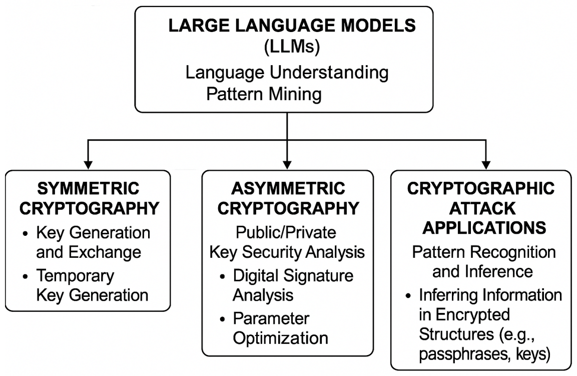
LARGE LANGUAGE MODELS (LLMs)
Language Understanding
Pattern Mining
SYMMETRIC CRYPTOGRAPHY
• Key Generation and Exchange
• Temporary Key Generation
ASYMMETRIC CRYPTOGRAPHY
Public/Private Key Security Analysis
• Digital Signature Analysis
• Parameter Optimization
CRYPTOGRAPHIC ATTACK APPLICATIONS
Pattern Recognition and Inference
• Inferring Information in Encrypted Structures (e.g., passphrases, keys)

Fig. 9. Large Language Modeling in Modern Cryptographic Systems

7 CONCLUSION

This paper examines in detail the practical application of large language models in the field of information security technology and demonstrates the strong potential and practical value of large model technology for key issues such as malware detection, network security analysis, and security vulnerability detection. Looks forward to the future security areas where large language models may be applied, such as the field of cryptography and malicious code detection. It is shown that the use of large language models can significantly improve the detection accuracy of security threats, reduce false alarms, and speed up the threat identification and response. Experimental results validate the efficiency of these models in understanding complex security data, highlighting their superior ability to capture malicious behavior patterns as well as analyze security events.

Although the research in this paper has achieved certain results, we still need to face up to its shortcomings. In particular, the application scope and research depth of the large model are yet to be further expanded. Therefore, in our future research work, we should be committed to broadening the breadth of the study, digging deeper into the depth of the study, and continuously improving the height of the study, with a view to achieving more comprehensive and in-depth results. Given the limitations and challenges identified in the current study, future work will focus on the following areas:

- **Enhanced Model Transparency and Interpretability:** Enhance model interpretability by integrating interpretable AI mechanisms that enable security experts to better understand and trust the model's decision-making process.
- **Defending Against Adversarial Attacks:** Research and development of more robust macromodels to enhance their robustness in the face of attacks, including the design of new training methods for recognizing and confronting adversary-generated adversarial samples.
- **Enhancement of model generalization capability:** Explore ways to enhance cross-domain data applicability, e.g., using techniques such as transfer learning and multi-task learning to optimize the model's ability to generalize to different domains and types of security threats.
- **Automated Security Intelligence Collection and Analysis:** Integrate large language models to achieve fully automated threat intelligence collection, processing, and analysis, facilitating rapid sharing of security information and intelligent generation of response strategies.
- **Extended Cryptographic Applications:** Research on further applications of large language models within the field of cryptography, such as automatic generation of secure cryptographic strategies, optimization of encryption algorithms, etc.

REFERENCES

[1] Changxiang Shen, HuangGuo Zhang, Dengguo Feng, ZhenFu Cao, and JiWu Huang. Survey of information security. *Science in China Series F: Information Sciences*, 50(3):273–298, 2007.

[2] Zongwei Li, Wenkai Li, Xiaoqi Li, and Yuqing Zhang. Stateguard: Detecting state derailment defects in decentralized exchange smart contract. In *Companion Proceedings of the ACM Web Conference 2024*, pages 810–813, 2024.

[3] Mikko T Siponen and Harri Oinas-Kukkonen. A review of information security issues and respective research contributions. *ACM SIGMIS Database: the DATABASE for Advances in Information Systems*, 38(1):60–80, 2007.

[4] Wenkai Li, Xiaoqi Li, Yuqing Zhang, and Zongwei Li. Defitail: Defi protocol inspection through cross-contract execution analysis. In *Companion Proceedings of the ACM Web Conference 2024*, pages 786–789, 2024.

[5] Weibo Liu, Zidong Wang, Xiaohui Liu, Nianyin Zeng, Yurong Liu, and Fuad E Alsaadi. A survey of deep neural network architectures and their applications. *Neurocomputing*, 234:11–26, 2017.

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

[7] Zongwei Li, Wenkai Li, Xiaoqi Li, and Yuqing Zhang. Guardians of the ledger: Protecting decentralized exchanges from state derailment defects. *IEEE Transactions on Reliability*, 2024.

[8] Zhi-Hua Zhou. A brief introduction to weakly supervised learning. *National science review*, 5(1):44–53, 2018.

[9] Samira Pouyanfar, Saad Sadiq, Yilin Yan, Haiman Tian, Yudong Tao, Maria Presa Reyes, Mei-Ling Shyu, Shu-Ching Chen, and Sundaraja S Iyengar. A survey on deep learning: Algorithms, techniques, and applications. *ACM computing surveys (CSUR)*, 51(5):1–36, 2018.

[10] Ting Chen, Xiaoqi Li, Xiapu Luo, and Xiaosong Zhang. System-level attacks against android by exploiting asynchronous programming. *Software Quality Journal*, 26(3):1037–1062, 2018.

[11] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 1(2), 2023.

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

[13] G Bharathi Mohan, R Prasanna Kumar, P Vishal Krishh, A Keerthinathan, G Lavanya, Meka Kavya Uma Meghana, Sheba Sulthana, and Srinath Doss. An analysis of large language models: their impact and potential applications. *Knowledge and Information Systems*, 66(9):5047–5070, 2024.

[14] Yuanzheng Niu, Xiaoqi Li, Hongli Peng, and Wenkai Li. Unveiling wash trading in popular nft markets. In *Companion Proceedings of the ACM Web Conference 2024*, pages 730–733, 2024.

[15] Enkelejda Kasneci, Kathrin Seßler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, et al. Chatgpt for good? on opportunities and challenges of large language models for education. *Learning and individual differences*, 103:102274, 2023.

[16] Yongchao Zhong, Bo Yang, Ying Li, Haonan Yang, Xiaoqi Li, and Yuqing Zhang. Tackling sybil attacks in intelligent connected vehicles: a review of machine learning and deep learning techniques. In *2023 8th International Conference on Computational Intelligence and Applications (ICCIA)*, pages 8–12. IEEE, 2023.

[17] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *ACM computing surveys (CSUR)*, 54(10s):1–41, 2022.

[18] Nazrul Hoque, Monowar H Bhuyan, Ram Charan Baishya, Dhruba K Bhattacharyya, and Jugal K Kalita. Network attacks: Taxonomy, tools and systems. *Journal of Network and Computer Applications*, 40:307–324, 2014.

[19] 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.

[20] Sven Ehlert, Dimitris Geneiatakis, and Thomas Magedanz. Survey of network security systems to counter sip-based denial-of-service attacks. *computers & security*, 29(2):225–243, 2010.

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

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

[23] Mohammed Abdulaziz Al Naeem, Adamu Abubakar, and MM Hafizur Rahman. Dealing with well-formed and malformed packets, associated with point of failure that cause network security breach. *Ieee Access*, 8:197554–197566, 2020.

[24] Mamoun Alazab. Profiling and classifying the behavior of malicious codes. *Journal of Systems and Software*, 100:91–102, 2015.

[25] 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.

[26] Martin E Hellman. An overview of public key cryptography. *IEEE Communications Magazine*, 40(5):42–49, 2002.

[27] Zekai Liu, Xiaoqi Li, Hongli Peng, and Wenkai Li. Gastrace: Detecting sandwich attack malicious accounts in ethereum. In *2024 IEEE International Conference on Web Services (ICWS)*, pages 1409–1411. IEEE, 2024.

[28] Yanchen Qiao, Weizhe Zhang, Xiaojiang Du, and Mohsen Guizani. Malware classification based on multilayer perception and word2vec for iot security. *ACM Transactions on Internet Technology (TOIT)*, 22(1):1–22, 2021.

[29] Sharoug Alzaidy and Hamad Binsalleeh. Adversarial attacks with defense mechanisms on convolutional neural networks and recurrent neural networks for malware classification. *Applied Sciences*, 14(4):1673, 2024.

[30] Aryan Marwaha, Rami Qays Malik, Shehab Mohamed Beram, Ali Rizwan, Kakarla Hari Kishore, Deepak Thakur, Tanya Gera, and Mohammad Shabaz. Retracted: Visualisation-based binary classification of android malware using vgg16. *IET Software*, 17(4):717–728, 2023.

[31] Xiangfan Wu, Ju Xing, and Xiaoqi Li. Exploring vulnerabilities and concerns in solana smart contracts. *arXiv preprint arXiv:2504.07419*, 2025.

[32] Wenkai Li, Zhijie Liu, Xiaoqi Li, and Sen Nie. Detecting malicious accounts in web3 through transaction graph. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, pages 2482–2483, 2024.

[33] Minjong Cheon, Hyodong Ha, Ook Lee, and Changbae Mun. A novel hybrid deep learning approach to code generation aimed at mitigating the real-time network attack in the mobile experiment via gru-lm and word2vec. *Mobile Information Systems*, 2022(1):3999868, 2022.

[34] Michael Guastalla, Yiyi Li, Arvin Hekmati, and Bhaskar Krishnamachari. Application of large language models to ddos attack detection. In *International Conference on Security and Privacy in Cyber-Physical Systems and Smart Vehicles*, pages 83–99. Springer, 2023.

[35] Crispin Almodovar, Fariza Sabrina, Sarvnaz Karimi, and Salahuddin Azad. Can language models help in system security? investigating log anomaly detection using bert. In *Proceedings of the 20th Annual Workshop of the Australasian Language Technology Association*, pages 139–147, 2022.

[36] Liam Daly Manocchio, Siamak Layeghy, Wai Weng Lo, Gayan K Kulatilleke, Mohanad Sarhan, and Marius Portmann. Flowtransformer: A transformer framework for flow-based network intrusion detection systems. *Expert Systems with Applications*, 241:122564, 2024.

[37] Guangzhao Chai, Shiming Li, Yu Yang, Guohui Zhou, and Yuhe Wang. Ctsf: An intrusion detection framework for industrial internet based on enhanced feature extraction and decision optimization approach. *Sensors*, 23(21):8793, 2023.

[38] Angel Jones and Marwan Omar. Codesentry: Revolutionizing real-time software vulnerability detection with optimized gpt framework. *Land Forces Academy Review*, 29(1):98–107, 2024.

[39] Sowon Jeon, Gilhee Lee, Hyoungshick Kim, and Simon S Woo. Smartcondetect: Highly accurate smart contract code vulnerability detection mechanism using bert. In *KDD workshop on programming language processing*, volume 16, page 225, 2021.

[40] Zoltán Szabó and Vilmos Bilicki. A new approach to web application security: Utilizing gpt language models for source code inspection. *Future Internet*, 15(10):326, 2023.

[41] Wenkai Li, Xiaoqi Li, Zongwei Li, and Yuqing Zhang. Cobra: interaction-aware bytecode-level vulnerability detector for smart contracts. In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering*, pages 1358–1369, 2024.

[42] Constantinos Patsakis, Fran Casino, and Nikolaos Lykousas. Assessing llms in malicious code deobfuscation of real-world malware campaigns. *Expert Systems with Applications*, 256:124912, 2024.

[43] Brij B Gupta, Akshat Gaurav, Varsha Arya, Wadee Alhalabi, Dheyaaldin Alsalman, and P Vijayakumar. Enhancing user prompt confidentiality in large language models through advanced differential encryption. *Computers and Electrical Engineering*, 116:109215, 2024.