Research on Deep Learning and Feature Aggregation Techniques for Web Security

Jinxin Wang

School of Vocational Education and Training, Linyi Vocational College, Linyi 276017, China
E-mail: wangjinxin202412@163.com

Received 14 January 2025; Accepted 18 March 2025

Abstract

With the rapid development of internet technologies, Web services have been widely applied in various fields, including finance, healthcare, education, e-commerce, and the Internet of Things, bringing great convenience to humanity. However, Web security threats have become increasingly severe, with side-channel attacks (SCA) emerging as a covert and highly dangerous attack method. SCAs exploit non-explicit information, such as network traffic patterns and response times, to steal sensitive user data, posing serious threats to user privacy and system security. Traditional detection methods primarily rely on rule-based feature engineering and statistical analysis, but these methods show significant limitations in terms of detection performance when dealing with complex attack patterns and high-dimensional, large-scale network traffic data. To address these issues, this paper proposes a side-channel leakage detection method based on SSA-ResNet-SAN. The SSA (sparrow search algorithm) is an optimization mechanism, intelligently searching for globally optimal feature subsets to enhance the model's feature selection capabilities and global optimization performance. Combined with deep residual networks (ResNet) and the signature aggregation network (SAN), the method performs a comprehensive analysis of both single-attribute and aggregated-attribute features in network traffic, thereby improving the model's accuracy and

Journal of Web Engineering, Vol. 24_2, 291–316.
doi: 10.13052/jwe1540-9589.2426

robustness. Experimental results demonstrate that SSA-ResNet-SAN significantly outperforms existing methods on multiple practical datasets. On the Google dataset, the use of aggregated attribute features enables SSA-ResNet-SAN to achieve an accuracy of 93%, which is substantially higher than that of other models. Furthermore, in multi-class tasks on the Baidu and Bing datasets, SSA-ResNet-SAN exhibits strong robustness and applicability. These experimental results fully validate the outstanding performance of SSA-ResNet-SAN in side-channel leakage detection, providing an efficient and reliable solution for the field of Web security.

Keywords: Web security, network traffic analysis, deep learning, search engine vulnerability, cybersecurity.

1 Introduction

With the rapid development of internet technologies, Web services have deeply penetrated various aspects of social life, including finance, healthcare, education, e-commerce, and the Internet of Things, bringing unprecedented convenience to humanity [1–3]. However, along with this fast-paced technological advancement, the threat of cyberattacks has also increased, especially security incidents involving data theft and privacy leakage through Web service vulnerabilities. Among these, side-channel attacks (SCAs) have gained increasing attention in recent years as a subtle and highly threatening form of attack [4–6]. Unlike traditional attacks, side-channel attacks usually exploit non-explicit information from system operations (such as network traffic patterns, response times, packet sizes, etc.) to infer and extract sensitive user data. Due to their covert nature, low technical barriers, and significant potential for harm, side-channel attacks have become an important hidden risk affecting Web information security [7–9]. Research has shown that modern Web systems, particularly in scenarios requiring high-frequency user requests such as search engines and recommendation systems, are particularly vulnerable to side-channel attacks. These types of attacks can steal private data or infer sensitive system information without disrupting system operation, causing immeasurable damage to users and organizations [10–13]. Therefore, efficient detection and defense against Web side-channel attacks have become a research focus and technical challenge in the field of cybersecurity.

In addressing side-channel attacks, traditional detection methods primarily rely on rule-based feature engineering and statistical analysis techniques [14, 15]. However, as attack technologies become more complex and data

scales grow, these methods show significant limitations when facing large-scale, high-dimensional network traffic. In recent years, the rise of deep learning technologies has brought new insights to solving this problem [16]. As a data-driven approach, deep learning can automatically extract complex feature representations from massive data and capture hidden relationships between data through nonlinear modeling, making it widely applied in side-channel leakage detection and quantification [17, 18].

Existing studies demonstrate that deep learning methods have shown significant advantages in network traffic analysis, pattern recognition, and covert attack detection. For example, by using convolutional neural networks (CNNs) to extract spatial features from traffic data or recurrent neural networks (RNNs) to capture temporal dependencies in traffic sequences, researchers have been able to effectively detect complex side-channel attacks. Additionally, some deep learning-based quantification techniques can assess the severity of side-channel leaks, providing valuable guidance for system design and optimization. However, despite the great potential of deep learning methods, their performance in practical applications still faces certain limitations.

Although deep learning methods have made certain progress in side-channel leakage detection, there are still many challenges in current research [19–22]. First, existing methods often focus on analyzing single-dimensional or single-attribute features, neglecting the interaction between different attributes, which limits detection effectiveness. For instance, there might be a high correlation between packet size, response time, and traffic sequence features, but traditional methods often model these features independently, ignoring their interrelationships [23, 24]. Second, most research designs detection algorithms for general scenarios, but in specific application scenarios (such as search engine autocomplete functions), model performance often fails to meet expectations [25]. Side-channel attacks in search engine autocomplete scenarios are characterized by frequent data interactions and complex traffic patterns, which place higher demands on the real-time and accuracy of detection algorithms. Moreover, due to the high dimensionality, strong nonlinearity, and noise interference of network traffic data, existing deep learning methods often face issues of low computational efficiency and insufficient model robustness when dealing with large-scale network traffic data [26].

To address the aforementioned challenges, this paper introduces a side-channel leakage detection method based on SSA-ResNet-SAN. The SSA-ResNet-SAN method analyzes the attribute features of network traffic

packets, constructing both single-attribute and aggregated-attribute feature vectors from filtered traffic files to comprehensively capture the interaction among multidimensional features. Specifically, this approach integrates ResNet with SSA to facilitate accurate modeling and feature extraction of network traffic, thereby significantly enhancing detection accuracy and robustness. Additionally, SSA-ResNet-SAN incorporates targeted optimization strategies tailored to the specific requirements of the search engine autocomplete scenario. By effectively capturing pattern features and temporal dynamics within data traffic, it substantially improves the detection of side-channel vulnerabilities. Experimental results demonstrate that SSA-ResNet-SAN outperforms existing methods across several practical datasets, particularly in the search engine autocomplete context, showcasing exceptional detection performance and applicability.

2 Related Work

In the area of side-channel leakage detection for Web applications, researchers have proposed various methods, ranging from early traditional techniques based on rules and statistical analysis to modern machine learning and deep learning algorithms [27–29]. These methods have alleviated, to some extent, the security threats posed by side-channel attacks. Among the early approaches, some studies utilized round trip time (RTT) between networks as an attack feature, inferring users' browsing behavior by monitoring delay patterns when accessing target websites [30]. For instance, attackers successfully identified specific websites or Web applications a user was visiting by comparing differences in RTT [31]. Although these methods were effective in certain situations, they relied on manual feature extraction and had low robustness, being sensitive to noise interference, which limited their applicability. Subsequently, researchers began to incorporate machine learning techniques to improve detection performance by modeling and classifying coarse-grained features of network traffic [32, 33]. Classical classifiers such as support vector machines (SVM), random forests (RF), and K-nearest neighbors (KNN) have been widely applied to side-channel attack detection tasks. These methods are capable of uncovering pattern features from large amounts of network traffic, thus effectively distinguishing normal traffic from abnormal traffic. For example, by analyzing the traffic feature curves generated by users' keystrokes, some studies have used differential features to identify the identity of the user. However, traditional machine learning methods heavily rely on feature engineering, often requiring manual

design of data features, and they struggle to capture potential attack patterns in high-dimensional and complex scenarios [34, 35].

With the rise of deep learning, researchers have begun exploring more intelligent detection methods, utilizing deep neural networks to automatically extract complex pattern features [36, 37]. For example, stacked denoising autoencoders (SDAE) can extract key features from data with significant noise interference through unsupervised learning of network traffic. Convolutional neural networks (CNNs) excel at capturing spatial features of traffic data and can identify local patterns within side-channel data. Meanwhile, long short-term memory (LSTM) networks are adept at handling time-series features and enhance detection accuracy by capturing temporal correlations in the traffic. Studies have shown that these deep learning methods significantly outperform traditional machine learning approaches in terms of accuracy and detection capability [38].

In terms of specific tool development, Sidebuster is an early black-box analysis tool capable of detecting side-channel vulnerabilities in Web applications [39, 40]. It primarily relies on automated fuzz testing combined with behavior analysis of Web traffic to identify potentially vulnerable modules. However, Sidebuster also has notable drawbacks, such as poor adaptability to complex application scenarios, especially in real-time environments (e.g., search engine auto-suggestions), where its performance is limited. Additionally, methods based on Google Web Toolkit (GWT) have been proposed, which perform preliminary exploration of side-channel attacks through fast frontend feature extraction [41]. However, these methods generally suffer from insufficient support for specific scenarios, poor real-time performance, or weak model generalization.

Although existing detection methods have demonstrated good performance in certain scenarios, they still have several shortcomings [42–44]. For example, most methods struggle to comprehensively analyze the interactions between multi-dimensional features, especially since the high-dimensional nature of network traffic data is often overlooked. Moreover, existing methods are typically designed for general detection scenarios and lack optimization for specific contexts (e.g., search engine auto-suggestions), resulting in poor performance in complex scenarios. Additionally, due to the dynamic changes in traffic data and noise interference, traditional methods still face challenges in terms of robustness and real-time processing capabilities in large-scale data environments [45]. These issues provide important directions for further optimization of Web application side-channel leakage detection and serve as key drivers for the development of new methods [18, 46].

3 Method

3.1 Overall Framework

As illustrated in Figure 1, this paper introduces a novel detection framework based on SSA-ResNet-SAN, meticulously designed to address the side-channel leakage challenges in search engine auto-suggestion scenarios. The proposed method seamlessly integrates ResNet, SAN, and the SSA to analyze network traffic feature information from multiple perspectives. Initially, ResNet is employed to extract the core features of individual attributes within the network traffic, effectively capturing intricate spatial patterns and filtering out key influencing factors through successive residual layers. The SAN module then aggregates attribute feature vectors, fusing multi-dimensional traffic features to establish global dependencies, thereby enhancing the model's capability to model complex interactions between these features. Finally, the SSA module leverages the sparrow search algorithm (SSA) to dynamically optimize feature selection and parameter configuration, ensuring globally optimal feature selection and efficient model training. The "Discoverer" and "Follower" mechanisms of SSA, in conjunction with the introduced "Sentinel" dynamic adjustment function, significantly bolster the model's robustness and efficiency. Through this architecture, the proposed method offers an accurate means of identifying side-channel vulnerabilities in search engine auto-suggestion scenarios, thus providing a robust technical foundation for side-channel leakage detection in Web applications.

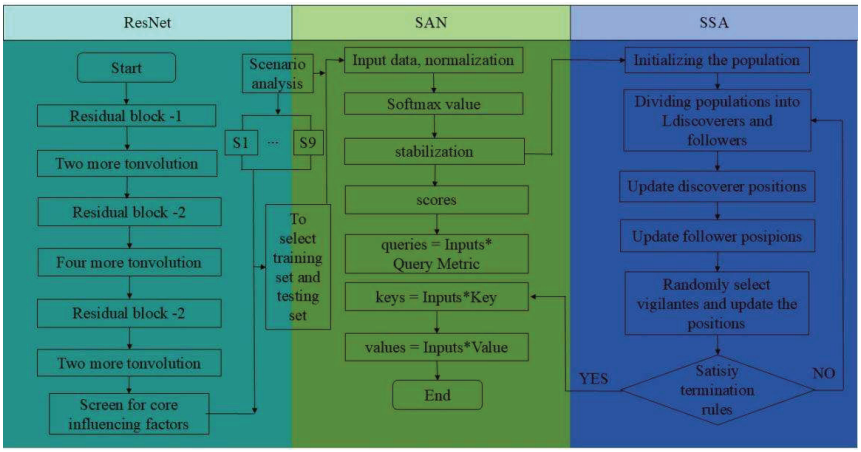

Figure 1 SSA-ResNet-SAN overall network architecture.

3.2 Sparrow Search Algorithm

The sparrow search algorithm (SSA) is a global optimization algorithm inspired by the cooperative behavior of sparrows in foraging, particularly the "discoverer–follower" collaboration strategy and the "sentinel" mechanism. SSA achieves a balance between global exploration and local exploitation for solving complex optimization problems. In SSA, each sparrow individual ($X_i = [x_{i1}, x_{i2}, \ldots, x_{id}]$) represents a d-dimensional solution vector, and the goal is to optimize the fitness function $f(X)$ through iterative updates. The population is initialized by randomly generating initial solutions, described as:

$$X_i(0) = X_{\min} + \mathrm{rand} \cdot (X_{\max} - X_{\min}) \tag{1}$$

where $X_{\min}$ and $X_{\max}$ represent the lower and upper bounds of the search space, and rand is a matrix of random numbers uniformly distributed in $[0, 1]$.

Discoverers are responsible for global exploration. Their position update strategy uses exponential decay and directional adjustments to balance exploration and exploitation:

$$X_i^{t+1} = X_i^t \cdot \exp\left(-\frac{i}{\alpha \cdot T}\right) + S \cdot (X_i^t - X_j^t) \cdot \mathbb{I}[R_1 \geq P_d] \tag{2}$$

where α is the scaling factor, T is the maximum number of iterations, $R_1 \sim U(0, 1)$ is a uniformly distributed random number, p_d is the alert threshold for discoverers, and $\mathbb{I}$ is an indicator function that determines whether directional adjustment is triggered.

Followers update their positions based on the behavior of the discoverers for local exploitation, described by:

$$X_i^{t+1} = X_i^t + F \cdot (X_{\mathrm{best}}^t - X_i^t) + \sigma \cdot \mathrm{rand} \tag{3}$$

where F is the learning rate, X_{best}^t represents the best solution in the current population, $\sigma \sim \mathcal{N}(0, \sigma^2)$ is Gaussian noise, and rand introduces random perturbations to enhance diversity.

The sentinel mechanism dynamically adjusts the positions of some individuals when the population is trapped in a local optimum. The sentinel updates are described as:

$$X_i^{t+1} = X_i^t + \beta \cdot \mathrm{sign}(f(X_{\mathrm{worst}}^t) - f(X_i^t)) \cdot \mathrm{rand} \tag{4}$$

where β is the adjustment coefficient, and $f(X_{\mathrm{worst}}^t)$ is the fitness value of the worst individual in the population.

The fitness optimization goal minimizes the distance between individuals and the global optimal solution, expressed in integral form as:

$$f(X_i^{t+1}) = \int_{\Omega} (X_i^{t+1} - X_{\text{opt}})^2 d\Omega \tag{5}$$

where X_{opt} represents the global optimal solution at the current iteration, and Ω denotes the search space.

To determine convergence, the mean change in the fitness of the population is used as the stopping criterion. The algorithm terminates when:

$$\frac{1}{N} \sum_{i=1}^{N} \left| f(X_i^t) - f(X_i^{t-1}) \right| < \epsilon \tag{6}$$

where N is the population size, and is the predefined threshold.

3.3 Signature Aggregation Network

The signature aggregation network (SAN) is a key module used to model and aggregate multidimensional features of network traffic. Its primary function is to allocate weights to features and produce global feature representations through the attention mechanism. Given the input data $X \in \mathbb{R}^{n \times d}$, where n represents the number of samples and d is the feature dimension, the SAN module first performs a linear transformation on the input data to generate queries Q, keys K, and values V. Specifically:

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V \tag{7}$$

where $W_Q, W_K, W_V \in \mathbb{R}^{d \times d'}$ are learnable weight matrices and d' is the transformed feature dimension. The queries Q and keys K are then used to compute the attention weight matrix A through dot product, followed by normalization:

$$A_{ij} = \frac{\exp\left(\frac{Q_i \cdot K_j^\top}{\sqrt{d'}}\right)}{\sum_{k=1}^{n} \exp\left(\frac{Q_i \cdot K_k^\top}{\sqrt{d'}}\right)}, \quad \forall i, j \in \{1, \ldots, n\}. \tag{8}$$

where $\sqrt{d'}$ is a scaling factor used to stabilize gradients. Using the weight matrix A, the value vectors V are aggregated through a weighted sum to generate the aggregated feature representation Z, expressed as:

$$Z_i = \sum_{j=1}^{n} A_{ij} V_j, \quad Z \in \mathbb{R}^{n \times d'}. \tag{9}$$

Finally, the aggregated features are mapped through a linear transformation and an activation function to produce the output ($Y \in \mathbb{R}^{n\times d}$), represented as:

$$Y = \sigma(ZW_O + b_O), \quad W_O \in \mathbb{R}^{d'\times d}, b_O \in \mathbb{R}^d, \tag{10}$$

where $\sigma(\cdot)$ denotes the activation function, such as ReLU or Sigmoid. Combining the above steps, the complete SAN process can be formulated as:

$$Y = \sigma\left(\left(\text{Softmax}\left(\frac{XW_Q(XW_K)^\top}{\sqrt{d'}}\right)XW_V\right)W_O + b_O\right). \tag{11}$$

SAN uses this mechanism to perform weighted aggregation and global modeling of features, effectively capturing complex interactions between features and providing richer feature representations for subsequent tasks. In this paper, the SAN module is used to further process the single-attribute features extracted by ResNet, producing aggregated features that significantly enhance the detection capability for side-channel leakage.

3.4 Data Collection for Side-channel Leakage Detection

To construct a high-quality dataset for side-channel leakage detection, this study utilizes a Docker container-based experimental environment in conjunction with the tcpdump tool for real-time network traffic capture. This approach ensures the precise recording of traffic characteristics during client–server interactions, thereby providing a comprehensive and accurate representation of network behavior. By simulating realistic search engine usage scenarios, the method generates network traffic data that is both highly precise and complete, facilitating robust side-channel leakage detection in a controlled yet dynamic environment. This meticulous data collection process forms the foundation for training and evaluating the proposed detection model, ensuring its relevance and effectiveness in real-world applications.

First, the experimental environment employs lightweight Docker containers, which offer excellent isolation, fast startup, and minimal resource consumption, providing a stable environment for monitoring and capturing network traffic. The Firefox browser running inside the container is used as the experimental tool to load the target search engine and open the search box. During the process of inputting a secret value, each character entered triggers an interaction between the client and the server, resulting in corresponding network traffic. To ensure accurate network traffic capture, the tcpdump tool is started before the input operation and is configured to monitor the

network interfaces inside the container. tcpdump is a powerful network traffic capturing tool capable of real-time packet capture and recording attribute information. Each secret value W consists of multiple characters and can be represented as: $(Y \in \mathbb{R}^{n \times d})$.

$$l_j \in \{A - Z, a - z, 0 - 9, \ldots\}, \quad j \in [1, r]. \tag{12}$$

During the input of a secret value, every character entry triggers one or more client–server interactions, generating one or more data packets. Once the secret value has been fully entered, tcpdump generates a corresponding network traffic file f, named after the secret value, which serves as a label for its associated traffic. Each generated traffic file f consists of multiple packets p, each recording attributes such as arrival time (time), destination address (Destination), source address (Source), protocol type, and packet size (length). Thus, a complete network traffic file f can be represented as:$f_i = \{p_1, p_2, \ldots, p_m\}$, $i \in [1, n]$, and the entire network traffic dataset can be represented as:

$$F = \{f_1, f_2, \ldots, f_n\}, \tag{13}$$

where F is generated by entering multiple secret values w_i during the experiment.

To ensure the proper functioning of the detection algorithm and enhance the reliability of the data, multiple sets of traffic data are collected for each secret value. In this way, the traffic data corresponding to each secret value w_i not only reflects traffic characteristics under different input conditions but also increases the diversity and adaptability of the dataset.

During the data collection process, the range and format of the secret values can be customized, such as restricting the number of characters or increasing complexity (e.g., mixed case, inclusion of special symbols), to generate traffic data with diverse characteristics. By controlling the input rhythm (e.g., character input intervals) and environmental variables (e.g., network delay and server response time), the dataset can be further enriched to cover a broader range of usage scenarios.

Ultimately, the collected network traffic dataset provides a solid foundation for subsequent side-channel leakage detection algorithms. The dataset contains rich packet-level attributes (e.g., timestamps, addresses, protocols, sizes), enabling detection models to extract features and train effectively, thereby helping the models identify side-channel leakage behaviors in search engine usage scenarios. Through precise experimental design and data collection methods, this study constructs a high-quality network traffic dataset that strongly supports research into side-channel leakage detection.

3.5 Data Noise Filtering Strategies for Side-channel Leakage Detection

In the task of side-channel leakage detection, the quality of network traffic data directly affects the performance of detection algorithms. However, in real-world network environments, there is often a large amount of irrelevant or invalid noise data, which can interfere with the feature extraction process and reduce detection accuracy. To address this issue, this paper proposes three noise filtering strategies based on protocol filtering, IP address filtering, and packet size filtering. These strategies effectively remove irrelevant traffic, optimize dataset quality, and provide a more reliable foundation for subsequent detection model training and testing.

The first step in enhancing the effectiveness of the dataset is noise filtering based on the protocol. In the experiment, key data transmission between the client and server primarily relies on the TCP protocol, with data encrypted using TLSv1.3 and TLSv1.2 protocols. Therefore, during the filtering process, all non-TCP traffic (such as UDP, ICMP packets, etc.) is discarded, leaving only reliable TCP data transmission traffic. Further inspection of the TCP packets is then performed to determine whether the TLS protocol is enabled, retaining only the encrypted packets that use TLSv1.3 and TLSv1.2. This approach significantly reduces the interference from plaintext communication or non-encrypted traffic, ensuring that the dataset contains only encrypted transmission-related data and providing higher-quality traffic data for side-channel attack research.

Second, noise filtering based on IP address can effectively remove background traffic that is irrelevant to the target scenario. In the experiment, the IP address of the target server is known, so by matching the source and destination IP addresses of the packets, only traffic related to the target server is retained. Specifically, all packets that match the target server's IP address are kept, while traffic from other unrelated IP addresses is discarded. This strategy significantly reduces background traffic, such as communication data between the client and other non-target servers, ensuring that only traffic relevant to the target scenario remains in the dataset. This IP address-based filtering method can effectively improve the purity of the data and reduce irrelevant noise during the training and testing processes. Finally, noise filtering based on packet size further cleans up anomalous or invalid packets. The experiment shows that many control packets (such as TCP handshake packets, heartbeat packets, or retransmission packets) and packets with abnormal lengths typically do not contain useful attack information and are considered

noise data. Therefore, by analyzing the normal packet size distribution of the target traffic, an effective length range $[L_{\min}, L_{\max}]$ can be determined. Any packets with lengths smaller than $L_{\min}$ or larger than $L_{\max}$ are labeled as invalid data and discarded. Additionally, for special packets within the boundary range, verification can be performed by examining protocol fields or traffic patterns in context, ensuring the accuracy and reliability of the filtering process. This strategy further reduces interference in the dataset and increases the proportion of valid data.

3.6 Construction of Aggregated Attribute Feature Vectors in Network Traffic

In side-channel leakage detection, constructing aggregated attribute feature vectors is a critical step to accurately describe the interaction behavior between the client and server. By processing and segmenting raw traffic packets into blocks and extracting statistical features of these blocks, aggregated attribute feature vectors transform fine-grained network activity data into high-dimensional representations suitable for model analysis.

During data collection, each time a user enters a character l_j in the search box, a burst of network activity is triggered between the client and server. These activities can be analyzed and segmented using time intervals ($\Delta = \{\Delta_1, \Delta_2, \ldots, \Delta_{p-1}\}$), where the interval is defined as: $\Delta_i = P_{i+1}.\mathrm{time} - P_i.\mathrm{time}$. When consecutive time intervals $(\Delta_1, \Delta_2, \ldots, \Delta_{i-1})$ are smaller than a threshold t, but $\Delta_i > t$, the sequence $([P_1, P_2, \ldots, P_i])$ is considered a complete data block B_1, while packet P_{i+1} belongs to the next block. Based on this rule, a network traffic file can be divided into multiple data blocks:

$$F = \{B_1, B_2, B_3, \ldots, B_s\}. \tag{14}$$

For each data block B_K, aggregated features can summarize the internal attributes of the block. For instance, the size feature of a data block can be defined as the total size of all packets within the block:

$$BS_k = \sum_{j=1}^{i_k} P_j.\mathrm{size} \tag{15}$$

where i_k is the number of packets in block B_K. Additionally, other features, such as the time span of packets in the block, protocol distribution, and the total number of packets, can be calculated to further enrich the representation. This aggregation approach not only reduces the complexity of fine-grained

individual packets but also captures global characteristics that reflect network behavior.

Finally, the aggregated features of a network traffic file can be represented as the set of features from all blocks:

$$\text{Feature}(F) = \{\text{Feature}(B_1), \text{Feature}(B_2), \ldots, \text{Feature}(B_s)\}. \quad (16)$$

These aggregated features not only describe the overall characteristics of network traffic but also reflect the changes in interaction behavior between the client and server, providing comprehensive and precise input for subsequent side-channel leakage detection. The construction of aggregated attribute feature vectors significantly improves the efficiency of network traffic analysis and enhances detection performance, laying a solid foundation for side-channel leakage detection in complex scenarios.

4 Experiments

4.1 Datasets

The experimental datasets used in this study for side-channel leakage detection are sourced from multiple search engines, covering various categories and task complexities to validate the effectiveness and applicability of the proposed methods. The initial experiments utilized data from Baidu and Bing search engines, with 10 designed categories, including news, games, actors, athletes, concerts, movies, and others. Each category contained 5 training samples and 5 testing samples, totaling 50 training samples and 50 testing samples, which were used to construct the basic 10-class classification task. Building on this, the classification task was further expanded to include 20-class and 30-class datasets. The 20-class dataset was created by pairing the 10 original categories, forming 5 groups. For example, Data Group 1 combined actors and athletes, while Data Group 2 combined movies and news. The 30-class dataset was created by grouping three categories at a time, resulting in 4 groups. For example, Data Group 1 combined actors, athletes, and concerts, while Data Group 2 combined games, deceased public figures, and recipes. This grouping approach increased the complexity of the classification task and was used to test the model's robustness across different classification scenarios. Furthermore, to validate the generalization capability of the proposed SSA-ResNet-SAN method and the LSTM-based side-channel leakage detection algorithm, the Google dataset was introduced for evaluation. The Google dataset consisted of the top 10 global trending

Table 1 The top 10 Google trending searches of the year

No.	Search Item	No.	Search Item
1	Hurricane Irma	6	Mayweather vs McGregor Fight
2	Matt Lauer	7	Solar Eclipse
3	Tom Petty	8	Hurricane Harvey
4	Super Bowl	9	Aaron Hernandez
5	Las Vegas Shooting	10	Fidget Spinner

search terms of 2017 as secret values, covering frequently searched topics across various domains (e.g., "hurricane," "bitcoin," "World Cup"). Table 1 shows the Top 10 Google Trending Searches.

4.2 Experimental Environment

To validate the effectiveness of the proposed side-channel leakage detection method, the experiment was designed with a high-performance software and hardware configuration to ensure the scientific rigor and reproducibility of the results. The software environment is based on the Ubuntu 20.04 LTS operating system, combined with Docker containers to provide an isolated experimental environment. Firefox browser (version 89.0) was used to simulate user search behavior. tcpdump 4.9.3 was employed to capture network traffic data, including timestamps, source addresses, destination addresses, protocol types, and packet sizes. Data preprocessing and feature extraction were performed using Python 3.8 and its main data processing libraries, such as Scapy, NumPy, and Pandas. The deep learning framework PyTorch 1.9.0 was utilized to implement the training and testing of the side-channel leakage detection models, including ResNet, SAN, and SSA algorithms. Additionally, the experimental scenarios included Google Search and DuckDuckGo, simulating high-traffic and privacy-preserving environments, respectively.

For the hardware environment, the experiment used an Intel Xeon E5-2680 v4 (14 cores, 2.4 GHz) multi-core processor to deliver robust computational performance, while an NVIDIA Tesla V100 GPU (32 GB HBM2 memory) accelerated the training of deep learning models. A 128 GB DDR4 memory was configured to handle large-scale network traffic data processing, and a 2 TB NVMe SSD ensured high-speed data storage and access. The network environment was configured with a 1 Gbps dedicated network, simulating standard latency (10–50 ms) and jitter (2–10 ms) conditions to reflect real-world usage scenarios. This software and hardware setup provided a reliable foundation for validating the proposed side-channel leakage detection

method and was capable of handling complex traffic analysis and model optimization requirements.

4.3 Evaluation Metrics

The side-channel leakage detection task in this study is framed as a multi-class classification problem. To evaluate the model's performance, several metrics are employed, including accuracy, precision, recall, and F1-score, complemented by three aggregation strategies: macro average, weighted average, and micro average.

Macro average is a balanced metric that does not consider differences in sample sizes between classes. It calculates the arithmetic mean of precision, recall, and F1-score across all classes. The formula for the macro-averaged precision is:

$$\text{Precision}_{\text{macro}} = \frac{\sum_{i=1}^{C} \text{Precision}_i}{C}, \tag{17}$$

where C is the total number of classes, and Precision_i represents the precision for the ith class. Similarly, the formulas for macro-averaged recall and F1-score are:

$$\begin{aligned} \text{Recall}_{\text{macro}} &= \frac{\sum_{i=1}^{C} \text{Recall}_i}{C}, \\ F1_{\text{macro}} &= \frac{\sum_{i=1}^{C} F1_i}{C}. \end{aligned} \tag{18}$$

Weighted average assigns weights to each class based on the proportion of samples in that class relative to the total number of samples, reflecting the model's performance more effectively when class distributions are imbalanced. The formula for weighted precision is:

$$\text{Precision}_{\text{weighted}} = \frac{\sum_{i=1}^{C} m_i \cdot \text{Precision}_i}{N}, \tag{19}$$

where m_i is the number of samples in the ith class, and $N = \sum_{i=1}^{C} m_i$ is the total number of samples. Similarly, the weighted recall and F1-score are calculated as follows:

$$\begin{aligned} \text{Recall}_{\text{weighted}} &= \frac{\sum_{i=1}^{C} m_i \cdot \text{Recall}_i}{N}, \\ F1_{\text{weighted}} &= \frac{\sum_{i=1}^{C} m_i \cdot F1_i}{N}. \end{aligned} \tag{20}$$

Micro average calculates the metrics by constructing a global confusion matrix across the entire dataset, ignoring the class labels. It aggregates the true positives (TP), false positives (FP), and false negatives (FN) for all classes to compute precision, recall, and F1-score. The formulas for micro-averaged precision, recall, and F1-score are:

$$\begin{aligned}
\text{Precision}_{\text{micro}} &= \frac{\sum_{i=1}^{C} \text{TP}_i}{\sum_{i=1}^{C} (\text{TP}_i + (\textit{Zolfpour} - \textit{Arokhloetal.})_i)}, \\
\text{Recall}_{\text{micro}} &= \frac{\sum_{i=1}^{C} \text{TP}_i}{\sum_{i=1}^{C} (\text{TP}_i + \text{FN}_i)}, \\
F1_{\text{micro}} &= \frac{2 \cdot \text{Precision}_{\text{micro}} \cdot \text{Recall}_{\text{micro}}}{\text{Precision}_{\text{micro}} + \text{Recall}_{\text{micro}}}.
\end{aligned} \tag{21}$$

Here, TP_i, $(\textit{Zolfpour} - \textit{Arokhloetal.})_i$, and FN_i are the true positives, false positives, and false negatives for the ith class, respectively.

4.4 The Impact of Different Noise Filtering Strategies on the Performance of SSA-ResNet-SAN

The impact of different noise filtering strategies on the performance of SSA-ResNet-SAN was analyzed by evaluating the model's accuracy across various experimental settings. The experiments were conducted on two datasets, Baidu and Bing, covering classification tasks across multiple categories such as actors, athletes, games, and news. Each experiment employed a different noise filtering strategy to preprocess the datasets, aiming to remove irrelevant or redundant information and improve detection accuracy.

As shown in Figure 2, Figure 2(a) illustrates the accuracy of the three experiments on the Baidu dataset. Experiment 1, which utilized an effective noise filtering method, achieved high accuracy across most categories, with performance in categories such as movies and actors approaching or reaching 90%. Experiment 2 exhibited relatively lower accuracy, indicating that its filtering strategy reduced noise at the cost of losing some critical information. Experiment 3, which adopted a more lenient filtering strategy and included more irrelevant data, experienced a significant drop in accuracy, with consistently low performance across all categories.

Figure 2(b) presents the experimental results on the Bing dataset. Similarly, Experiment 1 demonstrated the highest accuracy, exceeding 85% in

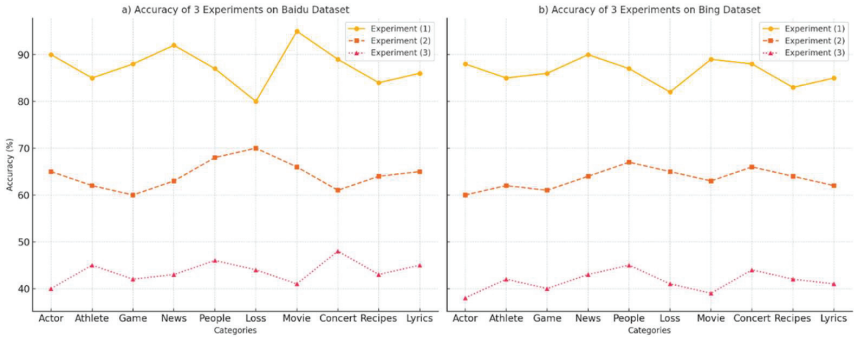

Figure 2 Accuracy results of three experiments conducted on the Baidu dataset (a) and the Bing dataset (b) across ten categories. Each line represents a different experimental setting, highlighting variations in performance across categories.

most categories. The results of Experiment 2 and Experiment 3 were consistent with those observed on the Baidu dataset, showing a certain degree of noise interference. Experiment 3, in particular, displayed significantly lower accuracy due to the inclusion of excessive noise. The consistency of results across the two datasets underscores the importance of noise filtering strategies in enhancing the performance of SSA-ResNet-SAN.

Overall, robust noise filtering strategies can effectively improve detection performance, whereas scenarios with high noise levels can significantly degrade the model's classification ability. Therefore, selecting appropriate preprocessing techniques is crucial to ensuring the SSA-ResNet-SAN model achieves superior performance in side-channel leakage detection tasks.

4.5 The Impact of Single-attribute Features and Aggregated-attribute Features on the Performance of SSA-ResNet-SAN

In this experiment, we compared the impact of single-attribute features and aggregated-attribute features on the performance of SSA-ResNet-SAN. The results are shown in Figure 3. Figure 3(a) presents the experimental results on the Baidu dataset. It can be observed that the detection accuracy of aggregated-attribute features is significantly higher than that of single-attribute features. In categories such as actors, athletes, games, and movies, the accuracy of aggregated-attribute features exceeds 95%, while the accuracy of single-attribute features is only between 85% and 90%. This

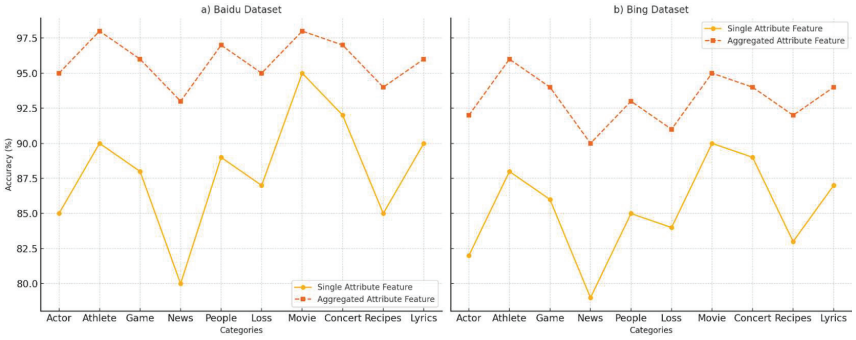

Figure 3 Accuracy comparison between single-attribute features and aggregated-attribute features on the Baidu dataset (a) and Bing dataset (b).

indicates that aggregated-attribute features can better capture the interactions between multi-dimensional attributes, thereby significantly improving detection performance.

Figure 3(b) shows the experimental results on the Bing dataset, exhibiting a similar trend to the Baidu dataset. The accuracy of aggregated-attribute features is higher than that of single-attribute features across all categories. For example, in categories such as athletes, concerts, and recipes, the accuracy of aggregated-attribute features approaches 95%, while the accuracy of single-attribute features is relatively lower, mostly ranging between 79% and 88%. This further validates the advantage of aggregated-attribute features in enhancing model performance.

Overall, on both the Baidu and Bing datasets, aggregated-attribute features significantly improve the classification accuracy of SSA-ResNet-SAN compared to single-attribute features. This demonstrates that in side-channel leakage detection tasks, leveraging aggregated multi-dimensional attribute features cffectively enhance the model's adaptability to complex data distributions and improve detection performance.

4.6 Experimental Results Analysis of the Side-channel Leakage Detection Method SSA-ResNet-SAN

As shown in Table 2, the accuracy of the Google dataset on the LSTM and SSA-ResNet-SAN models exhibits significant differences. When using single-attribute features, the accuracy of LSTM is 55%, while SSA-ResNet-SAN achieves 75%, which is significantly higher than that of LSTM. This indicates that SSA-ResNet-SAN has a stronger capability for side-channel

Table 2 Accuracy (%) of the Google dataset in LSTM and SSA-ResNet-SAN

	SSA-ResNet-SAN		LSTM	
Dataset/ Algorithm	Single-attribute Feature	Aggregated-attribute Feature	Single-attribute Feature	Aggregated-attribute Feature
Google	75	93	55	83

Table 3 Results of SSA-ResNet-SAN for aggregated-attribute features corresponding to each secret value (%)

Secret Value	Precision (%)	Recall (%)	F1-score (%)
Hurricane Irma	102	102	102
Matt Lauer	73	102	85
Tom Petty	85	102	93
Super Bowl	102	62	77
Las Vegas Shooting	102	102	102
Mayweather vs McGregor Fight	102	102	102
Solar Eclipse	102	102	102
Hurricane Harvey	102	102	102
Aaron Hernandez	73	102	85
Fidget Spinner	102	42	59

leakage detection when utilizing single-attribute features. In the case of aggregated attribute features, the accuracy of both models improves significantly. SSA-ResNet-SAN achieves an accuracy of 93%, which is 10 percentage points higher than the 83% accuracy of LSTM. This further validates that aggregated attribute features can effectively enhance detection performance, and it also demonstrates that SSA-ResNet-SAN has a stronger advantage in handling complex features and data interactions.

As shown in Table 3, the performance of SSA-ResNet-SAN for aggregated attribute features demonstrates strong overall results across the secret values. The model achieves perfect precision, recall, and F1 scores for several secret values, such as “Hurricane Irma,” “Las Vegas Shooting,” and “Mayweather vs McGregor Fight,” indicating its ability to effectively handle these categories. However, for certain secret values like “Super Bowl” and “Fidget Spinner,” the recall and F1 scores are relatively lower, suggesting challenges in capturing specific patterns for these categories. Despite these variations, the overall performance highlights the effectiveness of SSA-ResNet-SAN in leveraging aggregated attribute features for side-channel leakage detection, with consistently high precision and strong performance across the majority of the secret values.

5 Conclusions

This study addresses the issue of side-channel leakage detection in the realm of Web security and proposes an efficient detection method based on SSA-ResNet-SAN. By thoroughly analyzing the attribute features of network traffic, this approach constructs both single-attribute and aggregated-attribute feature vectors, integrating deep residual networks (ResNet) with a side-channel signature analysis mechanism (SSA). This combination significantly enhances the model's ability to capture interactions among multi-dimensional features. Furthermore, tailored optimization strategies are introduced to meet the specific demands of search engine autocomplete scenarios, improving both detection accuracy and adaptability. Experimental results demonstrate that SSA-ResNet-SAN outperforms existing methods across various practical datasets, particularly in complex Web interaction scenarios, highlighting its exceptional detection performance and potential for real-world applications.

Despite the progress made in Web side-channel leakage detection, there are still some limitations in this work. First, the computational complexity of the model when processing high-dimensional network traffic data is relatively high, which may impact its applicability in real-time scenarios with strict latency requirements. Second, the current experiments primarily rely on a limited dataset from search engine scenarios and have not fully covered more complex and diverse Web application environments. Furthermore, the model's performance under conditions of heavy noise interference or extreme data imbalance still requires further optimization, presenting challenges for detecting large-scale traffic in Web security.

Future research will focus on improving the adaptability and efficiency of the model in Web security scenarios. Specifically, the first direction is to optimize feature extraction and model architecture to reduce computational complexity and meet the high-performance demands of real-time Web interaction scenarios. Second, techniques such as joint learning and multi-task learning will be explored to enhance the model's generalizability across diverse Web contexts. Third, expanding the scale of the datasets by collecting real network traffic from a broader range of Web services will be a priority to evaluate the model's performance and stability in more complex Web security problems. Through these improvements, the proposed method aims to provide a more reliable and efficient solution for detecting side-channel attacks in Web security.

References

[1] I. Jemal, M. A. Haddar, O. Cheikhrouhou, and A. Mahfoudhi, "M-CNN: a new hybrid deep learning model for web security." pp. 1–7.

[2] Y. Pan, F. Sun, Z. Teng, J. White, D. C. Schmidt, J. Staples, and L. Krause, "Detecting web attacks with end-to-end deep learning," *Journal of Internet Services and Applications,* vol. 10, no. 1, pp. 1–22, 2019.

[3] M. Zhang, B. Xu, S. Bai, S. Lu, and Z. Lin, "A deep learning method to detect web attacks using a specially designed CNN." pp. 828–836.

[4] W. B. Shahid, B. Aslam, H. Abbas, S. B. Khalid, and H. Afzal, "An enhanced deep learning based framework for web attacks detection, mitigation and attacker profiling," *Journal of Network and Computer Applications,* vol. 198, pp. 103270, 2022.

[5] Y. Fang, Y. Li, L. Liu, and C. Huang, "DeepXSS: Cross site scripting detection based on deep learning." pp. 47–51.

[6] J. Wang, F. Li, S. Lv, L. He, and C. Shen, "Physically Realizable Adversarial Creating Attack against Vision-based BEV Space 3D Object Detection," *IEEE Transactions on Image Processing*, 2025.

[7] P. Yi, Y. Guan, F. Zou, Y. Yao, W. Wang, and T. Zhu, "Web phishing detection using a deep learning framework," *Wireless Communications and Mobile Computing,* vol. 2018, no. 1, pp. 4678746, 2018.

[8] J. C. Eunaicy, and S. Suguna, "Web attack detection using deep learning models," *Materials Today: Proceedings,* vol. 62, pp. 4806–4813, 2022.

[9] J. Wang, F. Li, and L. He, "A Unified Framework for Adversarial Patch Attacks against Visual 3D Object Detection in Autonomous Driving," *IEEE Transactions on Circuits and Systems for Video Technology*, 2025.

[10] Z. Tian, C. Luo, J. Qiu, X. Du, and M. Guizani, "A distributed deep learning system for web attack detection on edge devices," *IEEE Transactions on Industrial Informatics,* vol. 16, no. 3, pp. 1963–1971, 2019.

[11] D. S. Berman, A. L. Buczak, J. S. Chavis, and C. L. Corbett, "A survey of deep learning methods for cyber security," *Information,* vol. 10, no. 4, pp. 122, 2019.

[12] S. Toprak, and A. G. Yavuz, "Web application firewall based on anomaly detection using deep learning," *Acta Infologica,* vol. 6, no. 2, pp. 219–244, 2022.

[13] W. B. Shahid, B. Aslam, H. Abbas, H. Afzal, and S. B. Khalid, "A deep learning assisted personalized deception system for countering web

application attacks," *Journal of Information Security and Applications,* vol. 67, pp. 103169, 2022.

[14] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *Journal of Information Security and Applications,* vol. 50, pp. 102419, 2020.

[15] B. R. Dawadi, B. Adhikari, and D. K. Srivastava, "Deep learning technique-enabled web application firewall for the detection of web attacks," *Sensors,* vol. 23, no. 4, pp. 2073, 2023.

[16] F. Jiang, Y. Fu, B. B. Gupta, Y. Liang, S. Rho, F. Lou, F. Meng, and Z. Tian, "Deep learning based multi-channel intelligent attack detection for data security," *IEEE transactions on Sustainable Computing,* vol. 5, no. 2, pp. 204–212, 2018.

[17] W. Cui, T. Chen, and E. Chan-Tin, "More realistic website fingerprinting using deep learning." pp. 333–343.

[18] M. Li, and A. W. Guenier, "ChatGPT and Health Communication: A Systematic Literature Review," *International Journal of E-Health and Medical Communications (IJEHMC),* vol. 15, no. 1, pp. 1–26, 2024.

[19] L. Tang, and Q. H. Mahmoud, "A deep learning-based framework for phishing website detection," *IEEE Access,* vol. 10, pp. 1509–1521, 2021.

[20] D. Perdices, J. E. L. de Vergara, I. González, and L. de Pedro, "Web browsing privacy in the deep learning era: Beyond VPNs and encryption," *Computer Networks,* vol. 220, pp. 109471, 2023.

[21] H. Ran, W. Li, L. Li, S. Tian, X. Ning, and P. Tiwari, "Learning optimal inter-class margin adaptively for few-shot class-incremental learning via neural collapse-based meta-learning," *Information Processing & Management,* vol. 61, no. 3, pp. 103664, 2024.

[22] H. Zhang, C. Wang, L. Yu, S. Tian, X. Ning, and J. Rodrigues, "Pointgt: A method for point-cloud classification and segmentation based on local geometric transformation," *IEEE Transactions on Multimedia*, 2024.

[23] M. T. Muslihi, and D. Alghazzawi, "Detecting SQL injection on web application using deep learning techniques: a systematic literature review." pp. 1–6.

[24] Y. Zhou, Z. Wang, S. Zheng, L. Zhou, L. Dai, H. Luo, Z. Zhang, and M. Sui, "Optimization of automated garbage recognition model based on resnet-50 and weakly supervised cnn for sustainable urban development," *Alexandria Engineering Journal,* vol. 108, pp. 415–427, 2024.

[25] P. Yang, G. Zhao, and P. Zeng, "Phishing website detection based on multidimensional features driven by deep learning," *IEEE access,* vol. 7, pp. 15196–15209, 2019.

[26] H. Alkahtani, and T. H. Aldhyani, "Developing cybersecurity systems based on machine learning and deep learning algorithms for protecting food security systems: industrial control systems," *Electronics,* vol. 11, no. 11, pp. 1717, 2022.

[27] Y.-H. Choi, P. Liu, Z. Shang, H. Wang, Z. Wang, L. Zhang, J. Zhou, and Q. Zou, "Using deep learning to solve computer security challenges: a survey," *Cybersecurity,* vol. 3, pp. 1–32, 2020.

[28] R. Geetha, and T. Thilagam, "A review on the effectiveness of machine learning and deep learning algorithms for cyber security," *Archives of Computational Methods in Engineering,* vol. 28, no. 4, pp. 2861–2879, 2021.

[29] L. Zeng, "The Influence of Knowledge Worker Salary Satisfaction on Employee Job Performance," *Journal of Organizational and End User Computing (JOEUC),* vol. 35, no. 1, pp. 1–17, 2023.

[30] L. Lakshmi, M. P. Reddy, C. Santhaiah, and U. J. Reddy, "Smart phishing detection in web pages using supervised deep learning classification and optimization technique ADAM," *Wireless Personal Communications,* vol. 118, no. 4, pp. 3549–3564, 2021.

[31] J. Saxe, R. Harang, C. Wild, and H. Sanders, "A deep learning approach to fast, format-agnostic detection of malicious web content." pp. 8–14.

[32] A. Gaurav, B. B. Gupta, C.-H. Hsu, D. Perakoviæ, and F. J. G. Peñalvo, "Deep learning based approach for secure Web of Things (WoT)." pp. 1–6.

[33] Y. Li, and Z. Li, "Research on monitoring method of stock market systematic crash based on market transaction data," *Journal of Organizational and End User Computing (JOEUC),* vol. 35, no. 1, pp. 1–17, 2023.

[34] D. Chen, P. Wawrzynski, and Z. Lv, "Cyber security in smart cities: a review of deep learning-based applications and case studies," *Sustainable Cities and Society,* vol. 66, pp. 102655, 2021.

[35] N.-H. Nguyen, V.-H. Le, V.-O. Phung, and P.-H. Du, "Toward a deep learning approach for detecting php webshell." pp. 514–521.

[36] G. Apruzzese, M. Colajanni, L. Ferretti, A. Guido, and M. Marchetti, "On the effectiveness of machine and deep learning for cyber security." pp. 371–390.

[37] P. Abinaya, and S. J. Kumar, "Assured and provable data expuncturing in cloud using ciphertext policy-attribute based encryption (CP-ABE)," *CYBERNETICS AND SYSTEMS,* vol. 55, no. 4, pp. 786–803, 2024.

[38] Y. Shen, E. Mariconti, P. A. Vervier, and G. Stringhini, "Tiresias: Predicting security events through deep learning." pp. 592–605.

[39] A. Thakkar, and R. Lohiya, "A review on machine learning and deep learning perspectives of IDS for IoT: recent updates, security issues, and challenges," *Archives of Computational Methods in Engineering,* vol. 28, no. 4, pp. 3211–3243, 2021.

[40] S. Karthikeyan, and K. Lakshmi, "FPGA Based Integrated Control of Brushless DC Motor for Renewable Energy Storage System," *CYBERNETICS AND SYSTEMS*, 2023.

[41] Y. E. Seyyar, A. G. Yavuz, and H. M. Ünver, "An attack detection framework based on BERT and deep learning," *IEEE Access,* vol. 10, pp. 68633–68644, 2022.

[42] X. Wang, S. Wang, P. Feng, K. Sun, S. Jajodia, S. Benchaaboun, and F. Geck, "Patchrnn: A deep learning-based system for security patch identification." pp. 595–600.

[43] F. Y. Yavuz, "Deep learning in cyber security for internet of things," Fen Bilimleri Enstitüsü, 2018.

[44] T. M. Masenya, "Digital Transformation of Medical Libraries: Positioning and Pioneering Electronic Health Record Systems in South Africa," *International Journal of E-Health and Medical Communications (IJEHMC),* vol. 15, no. 1, pp. 1–13, 2024.

[45] A. Fidalgo, I. Medeiros, P. Antunes, and N. Neves, "Towards a deep learning model for vulnerability detection on web application variants." pp. 465–476.

[46] K. Ravikumar, P. Chiranjeevi, N. M. Devarajan, C. Kaur, and A. I. Taloba, "Challenges in internet of things towards the security using deep learning techniques," *Measurement: Sensors,* vol. 24, pp. 100473, 2022.

Biography

Jinxin Wang was born in Linyi, Shandong Province in 1986. He received a Bachelor of Arts degree from Linyi University in 2008 and a Master of Engineering degree from Shandong University in 2016. Since 2008, he has worked at Linyi Vocational College as a lecturer. His main research interests are information technology and software engineering. Since taking office, he has published 12 papers, 5 topics and participated in 1 Shandong teaching achievement award.