EAI Endorsed Transactions
on Scalable Information Systems
Research Article EAI.EU

Intelligent Detection Method for In-the-Wild Exploit Attacks in Distributed IoT Networks

Xiaohu Wu[1], Mingyuan Zhang[1], Xiaolei Liu[1,*] , Xiaojian Zhang[1] and Ke Jing[1]

[1]Jiangsu Electric Power Information Technology Co., Ltd., Nanjing 210000, China

Abstract

INTRODUCTION: In distributed IoT networks, massive devices, edge nodes and cloud platforms exchange data via open protocols, making device exploit attacks a severe threat to business continuity, data security and cross-domain propagation. Existing rule-based methods fail to balance real-time performance, accuracy and attack localization against rapidly mutating, obfuscated and large-scale in-the-wild exploits.
OBJECTIVES: To address the above limitations, this paper aims to propose an intelligent detection method integrating hybrid deep learning and open-source intelligence correlation for distributed IoT systems.
METHODS: First, HTTP packet preprocessing extracts suspected attack samples to mitigate extreme class imbalance. Second, a BERT-CNN coupled model classifies suspicious packets automatically. Finally, attack vector regression correlation with public vulnerabilities, PoCs and threat intelligence realizes accurate 1Day/NDay attack identification.
RESULTS: Experiments show the method achieves over 99.99% accuracy and 99.98% F1-score on real datasets. The IoT_Exploits_Founder system discovered 13 new in-the-wild exploits within one month in real environment. This study also supplements quantitative comparisons with representative methods, online deployment measurements of latency, memory overhead and throughput, and ablation studies for the attack-vector regression threshold and feature weights.
CONCLUSION: The proposed method provides effective support for AI-driven security monitoring in distributed IoT systems, with strong real-time edge applicability and robustness.

Keywords: IoT, Distributed Networks and Systems, In-the-Wild Exploitation, Threat Detection, OSINT

Received on 18 April 2026, accepted on 02 June 2026, published on 25 August 2026

doi: 10.4108/eetsis.12657

*Corresponding author. Email: stevencccccc@163.com

1. Introduction

Unlike traditional monolithic networks, new-generation distributed networks and systems are typically composed of massive IoT terminals, edge nodes, cloud platforms, and cross-domain data channels. Once IoT devices such as cameras, routers, NAS, and industrial control gateways are exposed to the Internet, they may become entry points for attackers to infiltrate distributed business systems, triggering cascading risks including service interruption, lateral propagation, and data asset damage. Existing studies have shown that IoT vulnerabilities feature complex types and a large attack surface, and large-scale exploitation activities on the Internet have already caused real-world harm [1].

Meanwhile, open-source information such as public vulnerability disclosures, Proof-of-Concept (PoC) codes, technical analyses, and social media discussions significantly accelerates the process of vulnerability weaponization [2-4]. Attackers can integrate publicly available exploit chains into automated attack scripts, botnet samples, or mass scanning tools within a short period, thereby launching large-scale, sustained in-the-wild exploit attacks against distributed IoT networks. Therefore, how to timely detect and identify vulnerability exploitation

behaviors targeting IoT devices in real network traffic has become a critical issue in security monitoring for distributed systems.

Existing detection methods are mainly divided into two categories: one relies on expert experience to construct rules or signatures, which are then deployed in intrusion detection systems [5-6]; the other adopts machine learning or deep learning for anomaly detection and traffic classification. However, the former lacks sufficient adaptability to unknown variants and rapidly evolving exploitation behaviors, while the latter often relies on closed or simulated datasets, mostly remaining at the binary "malicious/benign" classification stage, without the capability to accurately locate the categories of real in-the-wild exploits. For distributed IoT scenarios, binary alerts alone are often insufficient to support subsequent vulnerability analysis, asset inspection, and security operation decisions.

To address the above issues, this paper focuses on the demand for AI-driven threat detection in distributed IoT networks, and proposes an in-the-wild vulnerability exploit detection method based on hybrid deep learning classification and open-source intelligence correlation. Starting from real Internet traffic, this method realizes automatic detection and category identification of in-the-wild IoT vulnerability attacks through the collaboration of packet preprocessing, deep model classification, and vulnerability intelligence correlation. The main contributions of this paper are as follows:

1. A packet dataset for vulnerability exploits targeting real distributed IoT traffic scenarios is constructed. By systematically summarizing common in-the-wild IoT vulnerability attack paradigms, general detection rules are formed. Suspicious attack packets are extracted from real network traffic, and positive and negative samples are labeled with expert verification. The dataset contains 5×105 attack packets and 4×106 normal packets, which can provide a real-scenario verification foundation for related research.

2. A hybrid deep learning detection model fusing BERT and CNN is proposed. This model employs Transformer to learn global semantic representations of suspicious HTTP packets, and further combines CNN to extract local discriminative features, realizing automatic identification of in-the-wild vulnerability exploit packets. On the test set, the model achieves an F1-score of 99.98%, demonstrating high detection accuracy and practicality.

3. An open-source vulnerability intelligence correlation method based on attack vector regression is presented. This method extracts, standardizes and correlates public vulnerability disclosures, PoCs and related open-source threat intelligence, and integrates them with suspicious attack results output by the deep model, so as to achieve accurate localization of 1Day/NDay vulnerability attacks.

4. Based on the above methods, a real-time IoT vulnerability detection system named IoT_Exploits_Founder is implemented and deployed in a real-world environment for validation. Within a runtime of less than one month, the system discovered 13 new types of in-the-wild vulnerability attacks, showing that the proposed method can effectively support intelligent security monitoring and threat discovery in distributed IoT systems.

2. Related works

In essence, IoT in-the-wild vulnerability attack detection belongs to the problem of network intrusion detection and threat monitoring. However, in distributed IoT scenarios, its targets have expanded from traditional hosts and servers to massive heterogeneous terminals, edge nodes, and cross-domain communication links. Classic datasets such as KDD99 [7], ISCX 2012 [8], and CIDDS-001 [9] have promoted the development of anomaly detection-based research, enabling researchers to identify malicious activities from statistical features, protocol behaviors, and traffic structures. Early anomaly detection works mainly focused on statistical behavioral features, transmission control protocol characteristics, and traditional machine learning classifiers, such as hierarchical clustering, support vector machines, and naive Bayes [10–12]. Such methods are effective to a certain extent in scenarios with sufficient structured features and relatively stable attack patterns. Nevertheless, in distributed IoT networks, device types, protocol implementations, and packet formats are far more complex, and handcrafted features can hardly adapt to rapidly evolving vulnerability exploitation patterns.

With the development of mobile Internet and IoT technologies, research has shifted toward security monitoring in specific scenarios, such as malicious URL detection, Trojan traffic identification, malicious HTTP communication detection, and IoT intrusion detection [15–23]. Among them, deep learning methods show advantages in modeling complex traffic features. CNN, DNN, autoencoders, spatial-temporal models, and others have been widely used in malicious traffic detection and IoT security analysis [18–23]. However, most existing studies are still verified based on public benchmark sets or simulated traffic, which cannot fully reflect the variability, concealment, and category diversity of in-the-wild vulnerability attacks in the real Internet.

On the other hand, open-source information and network threat intelligence have been proven to provide important support for vulnerability exploitability assessment, attack analysis, and security alert verification [4, 24–25, 30]. Public vulnerability disclosures, PoCs, and technical analyses not only reveal vulnerability mechanisms but also provide critical clues for evidence extraction of attack behaviors and exploitation chain comparison. Yet in existing works, open-source intelligence is mostly used for manual analysis or post-incident judgment, with rare automatic integration with real-time traffic detection models.

In summary, existing research still has three shortcomings. First, there is a lack of large-scale vulnerability attack datasets targeting real distributed IoT network scenarios. Second, there is a lack of a unified

method that supports both automatic detection and attack type localization. Third, there is a lack of systematic research on the automatic correlation between deep learning classification results and open-source vulnerability intelligence. To address the above problems, starting from real Internet traffic, this paper proposes an in-the-wild vulnerability attack detection method for security monitoring of distributed IoT systems.

To make the positioning of this work clearer, this paper additionally evaluates representative rule-based IDS, traditional machine-learning classifiers, CNN/DNN-based IoT intrusion detection models, spatiotemporal deep-learning baselines, and the vanilla BERT model under a unified packet-level experimental protocol. This extended evaluation demonstrates that the proposed BERT-CNN structure improves not only the binary detection accuracy but also the practical trade-off among precision, recall, and real-time deployment overhead in distributed IoT monitoring scenarios.

3. Overview of the Detection Method

3.1. Analysis of the Current Situation of Vulnerability Attacks on IoT Devices

Vulnerability exploitation refers to a network intrusion behavior that takes advantage of defects in device hardware and software. Its primary objectives include gaining control of the device, implanting malware, stealing data, and so on. Vulnerability attacks have become one of the main means of network intrusion. Compared with traditional network devices, IoT devices are more likely to become targets of vulnerability attacks, mainly for two reasons. First, IoT devices extensively adopt various open-source components and services, which inherently contain numerous vulnerabilities. Second, IoT devices are usually deployed on a large scale, resulting in a massive number of devices with the same type of vulnerabilities exposed on the Internet, making them easy targets for various cybercriminals.

For instance, publicly disclosed vulnerabilities are often quickly absorbed and weaponized by automated attack tools, botnet samples, or reused PoC exploit chains within a short period [2–4, 24], as shown in Figure 1. It can be observed from the above cases that the time gap between vulnerability disclosure and large-scale exploitation is becoming increasingly narrow. Related studies also show that PoCs, technical analyses, and other public exploit clues significantly affect the likelihood of subsequent vulnerability development and exploitation [4, 24]. With the intensification of cyber attack and defense confrontations, timeliness and accuracy have become key technical indicators for IoT vulnerability attack detection methods. However, traditional detection methods that rely on expert experience for manual analysis and rule extraction can no longer meet the requirements of practical defense scenarios.

In vulnerability exploitation, there are three critical time points in the lifecycle from attack emergence to remediation: ① T_0: the moment when a vulnerability attack is first discovered; ② T_1: the moment when detailed information about the vulnerability attack is publicly disclosed; ③ T_2: the moment when the vendor releases a patch for the vulnerability. An attack occurring between T_0 and T_1 is defined as a 0Day vulnerability attack, during which no public information about the vulnerability exists online. An attack between T_1 and T_2 is referred to as a 1Day vulnerability attack. Attacks launched after T_2 are categorized as NDay vulnerability attacks. This paper mainly focuses on how to automatically and accurately identify various 1Day and NDay vulnerability attacks from massive network traffic. In contrast, 0Day vulnerability attacks cannot be automatically detected by the system and require a series of complex procedures such as manual verification and reproduction by domain experts.

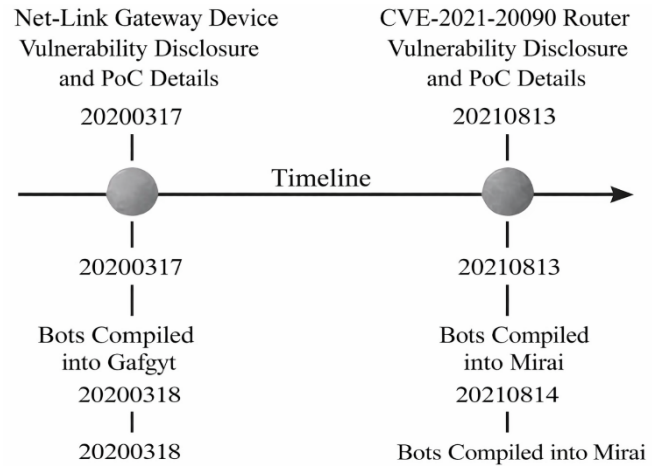

Figure 1. Timeline of IoT vulnerability information disclosure and in-the-wild exploit emergence. The legend clarifies vulnerability disclosure, informal exploit compilation, and botnet propagation events; T0, T1, and T2 denote first attack discovery, public disclosure, and vendor patch release, respectively

3.2. Framework of the Proposed Detection Method

The detection method in this paper is designed to meet the security monitoring requirements in distributed IoT networks and systems, and can detect and identify various in-the-wild vulnerability attacks targeting IoT devices from large-scale real network traffic. By integrating a hybrid deep learning model and open-source intelligence correlation techniques, this method minimizes reliance on manual rules while achieving real-time detection, accurate identification, and the capability of locating vulnerability types.

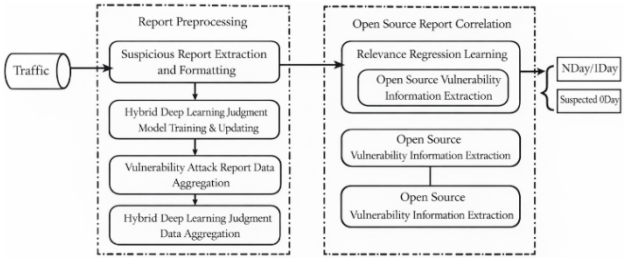

Figure 2. Overall framework of the proposed BERT-CNN and open-source-intelligence correlation detection method. The left dashed box denotes packet preprocessing and hybrid deep-learning judgement, while the right dashed box denotes vulnerability-report correlation and outputs NDay/1Day labels or suspected 0Day clues

The detection method proposed in this paper is composed of the following three modules: packet preprocessing, hybrid deep learning classification, and open-source intelligence correlation, as illustrated in Figure 2. The overall pipeline can be deployed in the traffic aggregation and security analysis links of distributed IoT networks, providing structured outputs for subsequent threat analysis and response.

1. Packet preprocessing. According to measurements on real traffic from a backbone network, IoT vulnerability exploit packets account for an extremely small proportion in actual Internet traffic (less than 0.01%), making vulnerability attack detection an extremely imbalanced classification problem. Directly feeding raw, unprocessed network traffic data into learning models for classification would result in a high false-negative rate. First, common paradigms of IoT vulnerability attacks are systematically summarized to formulate a set of general detection rules. Second, suspicious vulnerability attack packets are extracted from massive traffic based on these rules, which are used as input to alleviate the extreme class imbalance problem. Finally, the format and content of suspicious packets are standardized and field-trimmed, retaining only necessary information closely related to detection to meet the requirements of subsequent learning and correlation tasks.

2. Hybrid deep learning classification. Traditional detection methods based on generic feature extraction and simple machine learning models can hardly capture the key features of vulnerability attack packets accurately, nor adapt to the continuous evolution of attack patterns over time. Therefore, this paper proposes a hybrid deep learning model based on deep self-attention transformer networks and CNN. The model uses an attention mechanism to extract key semantic representations of packets, thereby realizing fast and automatic identification of attack packets. In the model training phase, the model is trained on a dataset containing 4×106 normal packets and 5×105 IoT vulnerability attack packets. In the inference phase, preprocessed suspicious attack packets are fed into the model to predict whether they are real vulnerability attack packets. In the model updating phase, newly confirmed attack packets are added offline to the training set for model retraining and updating.

3. Open-source intelligence correlation. Although suspicious packets classified as positive can be identified as IoT vulnerability attacks, it remains difficult to directly determine the specific vulnerability type, which requires open-source threat intelligence for precise identification. This paper tracks publicly available vulnerability disclosures, PoCs, and related threat intelligence sources rather than relying on a single website. These sources provide foundational information for exploit evidence extraction, exploitation pattern matching, and subsequent correlation recognition [24–25, 30]. By real-time crawling, cleaning, and standardization of exploit information from these sources, regression correlation is performed with positively classified attack packets. If the correlation is successful, the attack is identified as a 1Day or NDay vulnerability attack; otherwise, it may be a suspected 0Day attack that requires further manual analysis and confirmation.

In summary, this paper implements an end-to-end in-the-wild vulnerability attack identification method for distributed IoT systems, which can automatically detect, identify and determine various vulnerability attacks targeting IoT devices from background traffic, further accurately recognize the specific types of 1Day/NDay attacks, and uncover suspected 0Day attack clues at the same time. While improving the timeliness and accuracy of identification, the method reduces dependence on complete raw traffic through field truncation.

Finally, the IoT_Exploits_Founder system implemented based on this method ran continuously for one month in a real environment and discovered 13 new types of in-the-wild vulnerability attacks, which verifies its effectiveness in practical security monitoring scenarios.

4. Data Preprocessing

4.1. Attack Packet Extraction

This paper aims to detect IoT vulnerability exploit packets from real-time network traffic. However, in actual network traffic, the proportion of genuine IoT vulnerability attack packets is extremely small. Directly training and classifying real vulnerability attack packets from raw traffic using learning algorithms constitutes an extremely imbalanced classification problem, which is infeasible in practical scenarios. Therefore, data preprocessing is required to first extract all suspicious IoT vulnerability attack packets from the traffic; then, learning algorithms are used for training and classification, converting the extremely imbalanced learning problem into a conventional one. After data preprocessing, a large number of normal packets are filtered out, leaving only suspicious attack packets. The ratio of vulnerability attack packets

(positive samples) to normal packets (negative samples) is less than 1:105 before preprocessing, and is reduced to 1:10 after preprocessing, which greatly alleviates the dataset imbalance problem.

First, this paper analyzes the format types of numerous known IoT vulnerability attack packets and finds that standard HTTP requests are the only format for currently known IoT vulnerability attacks. By sending carefully crafted HTTP request packets to IoT devices, attackers can achieve the purpose of controlling and exploiting the devices. Therefore, in the extraction of suspicious attack packets, only packets in standard HTTP format are extracted. Second, after analyzing the malicious behaviors of various IoT vulnerability attack packets, it is found that vulnerability attacks targeting different IoT devices generally share common malicious behavioral characteristics and paradigms: ① Most attacks involve malicious behaviors such as remote command execution, malicious sample implantation, and behavior concealment, and these universal malicious features appear in the traffic packets of vulnerability attacks; ② Most vulnerable IoT device operating systems are based on the Linux kernel. Thus, these common malicious behaviors follow identical paradigms. For example, wget xxx; ./xxx; rm xxx represents a typical vulnerability attack behavior, including operations such as sample downloading, execution, and trace deletion.

Finally, this paper summarizes a complete set of general rules for IoT vulnerability exploitation behavioral characteristics, which can extract all suspicious IoT vulnerability attack packets from large-scale network traffic and greatly narrow the scope of vulnerability attack identification. However, the preprocessed packets still suffer from a high false positive rate and cannot accurately locate the category of vulnerability attacks. Therefore, deep learning classification and open-source intelligence correlation are required.

4.2. Packet Preprocessing

The suspicious vulnerability attack packets after extraction still require further format preprocessing. First, the character set of the packets is processed: various escape characters and computer-encoded characters in the packets are uniformly converted into text information in the standard printable character set. Second, the packet format is processed. In this paper, attack packets are limited to standard HTTP-format packets, which generally follow the structure shown in Figure 3.

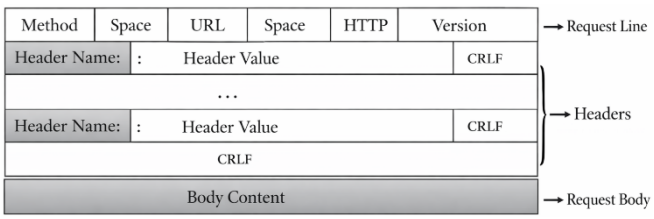

Figure 3. Typical HTTP request message format used for packet preprocessing

Among all known vulnerability attack packets, there are two types of request methods: HTTP GET and HTTP POST. The URL represents the access path of the targeted device and marks the entry point to programs or components on the device, making it one of the most critical fields in vulnerability attacks. If the request method is HTTP GET, the URL may carry parameters. For this reason, many vulnerability attacks exploit parameter entries in the URL. For instance, the CVE-2020-13872 attack shown in Figure 3 injects malicious commands by abusing URL parameters. If the request method is HTTP POST, the URL generally contains no parameters. During preprocessing of vulnerability attack packets, the URL is first extracted, followed by unified processing of variable parameter values.

Request header. Normal HTTP requests are usually sent by browsers, and their request headers and attribute values are generally filled automatically by the browser. In contrast, vulnerability attack packets are usually constructed and sent by attackers through custom attack scripts, and the header values in such packets are manually set by the attacker. There are no more than ten common types of request headers, most of which have little correlation with specific vulnerability exploitation. Therefore, in data preprocessing, most request header fields are directly discarded. Notably, some attackers insert special strings when filling request header information. For example, in the CVE 2021 20090 attack shown in Figure 4, the attacker filled the User-Agent field with the special string Dark, which can be used to trace and identify the attacker.

The main task of packet preprocessing is to separately extract the request method, processed URL field, partial request header fields, and request body fields from the packet, and directly concatenate them to obtain the suspicious attack packet text X, which serves as the input for subsequent learning-based classification and correlation recognition. This processing retains only necessary fields closely related to vulnerability exploitation detection, which helps reduce the interference of redundant content on model training and lowers the dependence of subsequent analysis on complete raw packets.

```
CVE_2021_20090
POST/images/../apply_abstract.cgi HTTP/1.1
Connection: keep-alive
User-Agent: Dark
action=start_ping&submit_button=ping.html&action_params=blink_time=
5&ARC_ping_ipaddress=212.192.241.7
%0AARC_SYS_TelnetdEnable=1&%0AARC_SYS_=cd+/tmp;wget xxxx;curl-O
xxx;chmod+777+lolol.sh;sh + lolol.sh&ARC_ping_status=0&TMP_Ping_Type=4
CVE_2020_13872
GET/portal/_ajax_explorer.sgi?action=umnt&path=path&where=here&en=;
cd+/tmp+rm +−
rf+*;wget+1.1.1.1;chmod+777+fetch.sh;sh+fetch.sh;HTTP/1.1
Connection: keep-alive
Accept-Encoding: gzip, deflate
Accept:^
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:60.0) Gecko/20100101 Firefox/60.0
```

Figure 4. Request message examples of two IoT exploit packets

5. Hybrid Deep Learning Detection and Classification

5.1. Introduction to the Vulnerability Attack Packet Dataset

Table 1. Partial In-the-Wild Exploited IoT Vulnerabilities in Recent 3 Years

CVE ID	Vulnerability Name	Targeted Device
CVE-2021-33544	UDP Technology Geutebruck IP Cameras Command Injection	IP Camera
CVE-2021-33514	HTTP Router Netgear GC108P	Netgear Router
CVE-2021-31755	Tenda AC11 Router Stack Buffer Overflow Vulnerability	Tenda Router
CVE-2021-28799	QNAP NAS Hybrid Backup Sync Command Injection	QNAP NAS Device
CVE-2021-20090	Arcadyan Buffalo Routers Configuration File Injection	Router with Arcadyan/Buffalo Components
CVE-2021-1497	Cisco HyperFlex HX Data Platform Command Execution	Cisco Management Platform
CVE-2020-9054	Zyxel NAS RCE Attempt Inbound	Zyxel NAS Device
CVE-2020-8949	Gocloud Router Remote Code Execution	Gocloud Router
CVE-2020-8515	DrayTek Vigor RCE	Vigor Router
CVE-2020-35713	Linksys RE6500 1_0_11_001 Remote Code Execution	Linksys Router
CVE-2020-35576	TpLink TLWR841N Command Injection	TpLink Router
CVE-2020-17456	Seowon Route	Seowon Router
CVE-2020-13872	DLink DIR 865L Ax120B01 Command Injection	Dlink Router
CVE-2020-10987	Tenda AC15 AC1900 goform setUsbUnload RCE	Tenda Router

The IoT vulnerability attack dataset used in this paper consists entirely of real packets captured at an Internet egress point from April to October 2021. All packets are in standard HTTP format and have undergone the data preprocessing operations described in Section 3, including both positive and negative samples.

Positive samples. There are 5×105 distinct HTTP packets in total, which are real IoT vulnerability attack packets detected and labeled from network traffic using rule-based and signature-based vulnerability attack detection rules developed by experts. These samples cover 197 types of IoT vulnerability attack behaviors. Table 1 lists some categories of in-the-wild IoT vulnerabilities exploited in the past three years. All these vulnerabilities are high-risk types that enable device control, such as remote command injection and arbitrary command execution. The affected device types mainly include various routers, cameras, and common components of related devices. Among them, 117 types have official CVE identifiers, while the remaining categories are manually verified and merged based on publicly disclosed exploitation information and PoC documents [24–25].

Negative samples. A total of 4×106 distinct HTTP packets are used as negative samples. Suspicious vulnerability attack packets are first filtered from network traffic using traffic extraction rules, and normal packets among them are then identified via a whitelist mechanism and labeled as negative samples in the dataset.

5.2. Hybrid Deep Learning Model

Reference [26] presents a systematic survey of deep learning models for text classification, showing that CNN, Transformer, and MLP are important basic architectures for such tasks. The model in this paper is a hybrid deep learning model combining BERT (bidirectional encoder representations from transformers) [29] and CNN [27], as illustrated in Figure 5. This model is used for classification experiments on vulnerability attack packets, where C

represents the contextual information of the text and T represents the individual token information of the text.

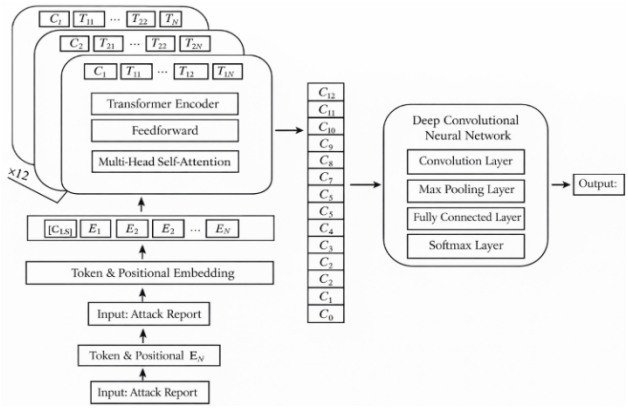

Figure 5. Structure of the proposed hybrid BERT-CNN classification model

BERT produces contextual representations from token and positional embeddings, and the multi-layer [CLS] vectors are fed into the CNN branch to extract local discriminative features for final classification.

In 2017, the Google team proposed the Transformer model [28]. Different from the traditional deep learning CNN model [27], this model directly employs the attention mechanism to process text, avoiding vector generation in a fixed sequence during text encoding, thus granting stronger parallel computing capability. In 2018, based on the Transformer model [28], the Google team introduced the BERT model [29], which uses a masked language model (MLM) for pre-training to produce deeper bidirectional representations. This model has achieved outstanding performance across various natural language processing (NLP) tasks.

When feeding textual feature vectors into the BERT model [29], a special token is prepended to the input sequence. The vector at the corresponding position in the final layer can serve as the semantic representation of the entire text for classification tasks. The self-attention module of BERT [29] enhances the semantic representation of a target word using other words in the text, while the original semantics of the target word remain dominant. Since the additional special token itself carries no inherent meaning, it can integrate semantic information from all words in the text more evenly than ordinary tokens, resulting in a better global semantic representation.

1. CNN model [27]. CNN was initially mainly used for image processing tasks. In 2014, Kim [27] proposed the TextCNN model, which applies the CNN model [27] to text classification. The CNN architecture adopted in this paper is shown on the right side of Figure 5. The maximum input passes through the convolution layer, max-pooling layer, fully connected layer, and softmax layer in sequence, and finally outputs the classification result. For the optimization of the CNN model [27], the ReLU activation function and Adam optimizer are used. To better capture local correlations and extract key information from text, three kernel sizes (3, 4, and 5) are selected.

2. Loss function. Cross-entropy is mainly used to measure the proximity between the actual output and the expected output. A smaller cross-entropy indicates a smaller discrepancy between them. Let p be the expected probability distribution and q the actual probability distribution; the cross-entropy is given by

$$H(p,q) = -\sum p(x) \ln q(x) . \tag{1}$$

3. Proposed model. Considering the textual characteristics of vulnerability attack packets, the model in this paper is designed to determine whether a suspicious attack packet is a real vulnerability attack. The model structure is shown in Figure 5. First, the text information X of each preprocessed suspicious vulnerability packet is treated as a single sentence, and is fed into the BERT model [29] after token embedding and position embedding. Next, the CLS vectors from the 12 hidden layers are concatenated as the input to the CNN model [27], which then goes through the convolutional layer, max-pooling layer, fully connected layer, and softmax layer. Finally, the classification result is output.

5.3. Evaluation Metrics

Following the common practice of evaluation metrics for classification algorithms, precision P, recall R, accuracy Acc, and F1-score are used to evaluate the classification performance of the hybrid deep learning model. Their formulas are given as follows:

$$R = T_P / (T_P + F_N) . \tag{2}$$

$$P = T_P / (T_P + F_P) . \tag{3}$$

$$A_{cc} = (T_P + T_N) / (T_P + T_N + F_P + F_N) . \tag{4}$$

$$F_1 = 2PR / (P + R) . \tag{5}$$

In the formulas, TP denotes true positives, FP denotes false positives, FN denotes false negatives, and TN denotes true negatives.

5.4. Classification Experiment Results and Analysis

The text X of suspicious vulnerability attack packets obtained after packet preprocessing is used as the input to the learning classification model to identify real IoT vulnerability attack packets. According to actual data statistics, the proportion of real vulnerability attack packets in suspicious packets ranges from 0.1 to 1. Therefore, four positive–negative sample ratios (1:1, 1:2, 1:4, and 1:8) are adopted for different classification experiments.

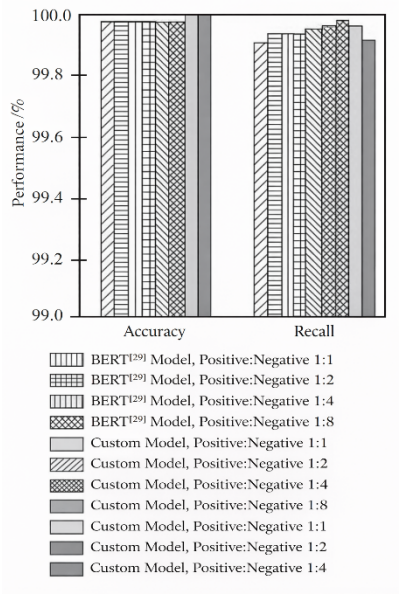

Figure 6. Precision and recall comparison under different positive-to-negative sample ratios. Each bar group corresponds to one model-ratio setting, and the legend distinguishes the vanilla BERT baseline from the proposed BERT-CNN model.

In the classification experiments, classification verification is performed using both the proposed model and the BERT model [29] for comparison, with the training set and test set split into 0.8 and 0.2 for each group. Under different positive–negative sample ratios and different model selections, the classification results on the test set are shown in Table 2 and Figure 6, respectively. Experimental results show that the proposed model achieves superior detection performance for vulnerability packet detection. First, the precision of both models is close to 1. In practical scenarios, precision P is a more practical metric, which measures how many labeled vulnerability attack packets are real attacks. Since classification is only one step in the proposed detection method, attack packets identified by the classification algorithm will proceed to subsequent processes for correlation and clue discovery. Low precision would negatively affect these downstream procedures. Second, the accuracy Acc and F1-score of both classification models exceed 99.9%. Especially under the positive–negative sample ratio of 1:2, the accuracy of the proposed model surpasses 99.99%, meaning that only a single-digit number of false positives occur per 105 suspicious packets detected on average, demonstrating strong practical value. Finally, as shown in Table 2, the F1-score of the proposed model is an average of 0.02% higher than that of the single BERT model [29]. This paper focuses on detecting vulnerability attacks in large-scale traffic, and the number of suspicious packets identified in real-world systems exceeds 108 per day. A 0.01% improvement in the F1-score means 104 additional packets are correctly identified, and vice versa, which is critical for discovering clues of NDay attacks and 0Day vulnerability attacks. Therefore, the 0.02% improvement has a significant impact on practical operations, greatly reducing false positives and enhancing the usability of the detection system. To further respond to the need for broader quantitative comparison, additional baselines were reproduced on the same preprocessed packet dataset, including a rule-based IDS, an SVM-based traditional machine-learning classifier, a CNN-based traffic detector, a DNN/autoencoder-based detector, and a spatiotemporal deep-learning model. The comparative results are summarized in Table 3.

Table 2. Experimental results of F_1 and A_{CC}

Positive-to-Negative Sample Ratio	F1/%		ACC/%	
	Bert[29] Model	Proposed Model	Bert[29] Model	Proposed Model
1:1 (Total: 106)	99.93	99.94	99.95	99.96
1:2 (Total: 1.5×106)	99.97	99.97	99.99	99.99
1:4 (Total: 2.5×106)	99.93	99.96	99.98	99.98
1:8 (Total: 4.5×106)	99.96	99.97	99.96	99.97
1:1 (Total: 106)	99.93	99.94	99.95	99.96

Table 3. Comparative results with representative IoT intrusion and malicious traffic detection methods on the same packet-level test set

Method	Main feature/model	Precision/%	Recall/%	ACC/%	F1/%

Rule-based IDS [5-6]	Expert signatures and protocol rules	96.38	87.62	94.21	91.78
SVM-based ML [11-12]	Handcrafted statistical and HTTP features	97.02	93.56	96.84	95.26
CNN-based IoT IDS [19-20]	Local packet-pattern convolution	99.58	99.36	99.62	99.47
DNN/autoencoder detector [21,23]	Deep feature reconstruction/classification	99.61	99.44	99.68	99.52
Spatiotemporal DL [22]	Temporal and spatial traffic representation	99.74	99.61	99.79	99.67
Vanilla BERT [29]	Global semantic representation	99.96	99.94	99.98	99.95
Proposed BERT-CNN	Global semantics + local discriminative features	99.98	99.97	99.99	99.97

As shown in Table 3, traditional rule-based and handcrafted-feature methods are more sensitive to exploit mutation and packet obfuscation, resulting in lower recall. Deep-learning baselines improve the detection capability, but the proposed BERT-CNN achieves the best overall F1-score because the BERT branch captures global semantic dependencies in HTTP exploit texts, whereas the CNN branch further enhances local command-fragment discrimination.

In addition, the training-loss records show that the proposed model converges more smoothly than the single BERT model [29]. Model training was accelerated using a GPU 4000 accelerator card. A comparison of training time per epoch is shown in Table 4. The training time of both models varies linearly with the number of samples, ranging from 60 to 170 minutes. After classifying suspicious vulnerability attack packets using the proposed model, although we can determine whether a packet is a real attack, we still cannot identify its specific attack type. Further correlation with open-source intelligence is required for precise vulnerability identification.

Table 4. Model training time in each round

Number of Samples/10^6	Training Time/min	
	Bert[29] Model	Proposed Model
1	62	71
1.5	89	102
2.5	127	146
4.5	146	167

5.5. Online Real-Time Deployment Evaluation

To verify the applicability of the proposed method for online security monitoring at resource-constrained IoT edge nodes, an additional deployment evaluation was conducted after model training. The test environment included a 4-core ARM-based edge gateway with 4 GB memory, and inference was performed on preprocessed suspicious HTTP packets. The evaluation considered single-packet inference latency, batch inference latency, peak memory overhead, and sustained throughput.

Table 5. Online deployment overhead of the proposed detection pipeline on an edge gateway

Deployment component	Latency per packet/ms	Peak memory/MB	Throughput/packets/s	Remark
Packet preprocessing	0.32	58	42,000	Rule filtering and field normalization
Vanilla BERT inference	6.84	890	146	Single suspicious packet
Proposed BERT-CNN inference	7.36	936	135	Single suspicious packet
Proposed BERT-CNN, batch=32	2.11	946	474	Batched online inference
Full pipeline, batch=32	2.56	988	391	Preprocessing + classification + correlation interface

The results in Table 5 show that the proposed model introduces only a limited additional memory overhead compared with vanilla BERT, while the batch-inference mode reduces the average per-packet latency to 2.11 ms. Considering that only suspicious packets extracted by preprocessing enter the deep model, the overall pipeline can satisfy near-real-time IoT exploit monitoring requirements on lightweight edge gateways.

6. Open-Source Information Correlation and Identification

6.1. Open-Source Vulnerability Information Extraction

In recent years, studies have shown that open-source information can provide important support for the generation of cyber threat intelligence, correlation analysis, and security decision-making [25, 30]. The open-source information concerned in this paper refers to vulnerability exploitation information (Exploit, EXP), i.e., detailed descriptions of how vulnerabilities are exploited or exploit code, which elaborate the mechanism and usage of vulnerabilities. Public PoCs, technical analyses, and exploitation clues can not only demonstrate the exploitability of vulnerabilities but also provide critical evidence for subsequent attack correlation and priority assessment [4, 24]. Vulnerability attack packets targeting various IoT devices specifically adopt methods from such publicly disclosed information, and security analysts also rely on such open-source intelligence to determine whether an attack packet belongs to a 1Day or NDay attack.

Such vulnerability exploitation information from public sources can be regarded as a form of open-source threat intelligence. However, its sources are scattered, description templates are inconsistent, and field granularity varies significantly [24–25, 30]. Besides vulnerability-related metadata such as release date and affected devices, the most critical information is the EXP content, namely the concrete exploitation scripts, which usually come in Shell, Python, PHP, Java, and other programming languages. Therefore, this paper conducts real-time tracking, crawling, and storage across multiple public vulnerability exploitation intelligence sources, rather than relying on a single web-based data source. As of October 2021, more than 2.5×105 entries of various vulnerability information have been collected from these public sources, covering diverse vulnerability types. Nevertheless, only a small proportion of them are 1Day or NDay vulnerabilities that can actually launch cyberattacks, perform command injection, or enable remote code execution (0Day vulnerability information is not publicly disclosed).

By further comparing and correlating the vulnerability attack packets detected in Section 5 with the open-source vulnerability information, the category of the attack packets can be confirmed. Different from previous manual comparison and correlation by security analysts, this paper proposes an open-source vulnerability intelligence correlation method based on attack vector regression, enabling accurate identification of vulnerability attack packets.

6.2. Vulnerability Correlation and Identification Based on Attack Vector Regression

Both vulnerability EXP information crawled from open-source vulnerability platforms and vulnerability attack packets in network traffic differ significantly from natural language short texts. Existing short text processing tools, such as NLTK and scikit-learn, mainly handle English and Western European languages, while HanLP is designed for Chinese text. Most functions in these tools cannot be directly applied to attack packets or EXP. This is because attack packets have no concept of a lexicon, and certain punctuation marks and individual characters are also part of code execution, so word segmentation cannot be directly performed based on spaces and punctuation. Therefore, this paper designs and implements a dedicated named entity recognition module to extract keyword sets separately from attack packets and EXP.

Information extraction is performed on vulnerability attack packets to extract attack vector fields related to vulnerability exploitation, forming the following structure: [x_1: attack_target; x_2: attack_para; x_3: attack_command]. Among them, the attack target refers to the specific address requested by the vulnerability attack packet, i.e., the specific path triggered when the device is attacked, which usually represents a service interface of a component or service on the device open to the outside, indicating the externally accessible location of the device. The attack parameter represents the parameter used to trigger the exploitation during a vulnerability attack. Table 6 lists the attack targets and attack parameters of the vulnerability attack packets mentioned in Table 1.

Table 6. Attack targets and parameters in IoT exploits

Vulnerability ID	Attack Target	Attack Parameters
CVE-2021-33544	/uapi-cgi/certmngr.cgi	action, creatSelfCert, local
CVE-2021-33514	cgi/setup.cgi	token

CVE-2021-31755	/goform/setmac	mac, wiffen, wifissid
CVE-2021-28799	/cgi-bin/backup/hbs_mgnt.cgi	run_cmd, jisoosocoolhbsmgnt
CVE-2021-20090	/images/../apply_abstract.cgi	action, submit_button, action_params, arc_ping_ipaddress
CVE-2021-1497	/storfs-asup	action, token
CVE-2020-9054	/cgi-bin/webLogin.cgi	adv, username, password
CVE-2020-8949	/cgi-bin/webui/admin/tools/app_ping/diag_ping/	None
CVE-2020-8515	cgi-bin/mainfunction.cgi	action, keyPath, loginPwd
CVE-2020-35713	/goform/setSysAdm	AuthTimeout, admPassHint
CVE-2020-35576	/cgi?	maxhopcount, numberoftries, host
CVE-2020-17456	/cgi-bin/system_log.cgi	Command, traceMode
CVE-2020-13872	/portal/__ajax_explorer.sgi	Action/path/where/en
CVE-2020-10987	/goform/setUsbUnload	setUsbUnload

Attack commands represent specific attack behaviors, such as sample implantation, trace hiding, trace deletion, and so on. Based on statistics of numerous known vulnerability attacks, common attack commands in IoT vulnerability attacks are listed in **Table 7**.

Table 7. Common attack commands in IoT exploits

Attack Command Type	Keywords
Sample Implantation	wget, curl, tftp, fetch
Reverse Shell Connection	bash -i, /tmp/socat exec
Probing	nc, echo, dnslog
Trace Hiding	rm, mv, base64, decodeHex
Command Execution	chmod, system, md5sum, busybox, exec

$$f(x,y) = w_1 \,\mathrm{sim}(x_1, y) + w_2 \,\mathrm{sim}(x_2, y) + w_3 \,\mathrm{sim}(x_3, y) + \alpha \quad (6)$$

$$f(x,y) = \begin{cases} >1 & \text{if } x \text{ and } y \text{ are corresponding} \\ <0 & \text{otherwise} \end{cases} . \quad (7)$$

In the formula, x stands for the text of the vulnerability attack packet, y represents the vulnerability exploitation information text in open-source vulnerability intelligence, x1, x2 and x3 are the three types of attack features directly extracted from the attack packet, w1, w2, w3 and α are correlation coefficients obtained through regression calculation between the attack packet dataset and open-source vulnerability information, and the sim() function indicates the similarity value of a certain attack type between the attack packet and open-source vulnerability information, ranging from 0 to 1, which is calculated as the proportion of keywords in a certain attack category of the attack packet that appear in the open-source vulnerability information; for example, for the vulnerability attack packet corresponding to CVE-2021-20090, its attack target is "/images/../apply_abstract.cgi", which is split into different strings by the path separator '/', and sim(x1,y) is 1 if all these strings appear in y and 0 if none of them appear in y, while the calculation methods of sim(x2,y) and sim(x3,y) are similar.

By obtaining w1, w2, w3, and α through regression on real-world data, the correlation function f(x,y) between attack packets and open-source vulnerability information can be constructed. In practical application, when an attack packet x needs to be correlated, it will be matched against all known open-source vulnerability records. The criteria for determining a successful correlation are as follows: first, select the vulnerability type y corresponding to the maximum value of f(x,y); second, check whether the value of f(x,y) for this y exceeds a predefined threshold. If it does, the attack packet is successfully correlated to vulnerability type y. This threshold is usually adjusted by experts according to actual deployment scenarios. Accordingly, the default values w1=0.48, w2=0.34, w3=0.18, α=0.03 and τ=0.70 are selected according to the ablation results in Tables 8 and 9, rather than being manually fixed without validation.

6.3. Ablation Analysis of Correlation Threshold and Regression Weights

The open-source intelligence correlation module depends on the correlation threshold and the regression weights of attack target, attack parameter, and attack command. To verify the rationality and robustness of these settings, this subsection adds two ablation experiments using manually verified 1Day/NDay exploit packets and matched public vulnerability intelligence records.

Table 8. Ablation results under different correlation thresholds

Threshold tau	Precision/%	Recall/%	F1/%	False correlation rate/%
0.50	98.86	99.51	99.18	1.14
0.60	99.21	99.34	99.27	0.79
0.70	99.63	99.18	99.40	0.37
0.80	99.82	98.93	99.37	0.18
0.90	99.91	96.76	98.31	0.09

As shown in Table 8, a low threshold improves recall but increases false correlations, whereas an overly high threshold reduces recall because some obfuscated exploit packets cannot be matched to open-source descriptions. Therefore, tau=0.70 is selected as the default threshold, achieving a balanced F1-score and a low false correlation rate.

Table 9. Ablation results of attack-vector regression weights

Weight setting	w1 target	w2 parameter	w3 command	alpha	F1/%
Target only	1.00	0.00	0.00	0.00	97.64
Parameter only	0.00	1.00	0.00	0.00	96.87
Command only	0.00	0.00	1.00	0.00	94.72
Equal weights	0.33	0.33	0.33	0.00	98.79
Without target feature	0.00	0.55	0.45	0.03	98.21
Without parameter feature	0.62	0.00	0.38	0.03	98.53
Without command feature	0.58	0.42	0.00	0.03	98.91
Regression-based setting	0.48	0.34	0.18	0.03	99.40

Table 9 indicates that attack target information contributes most to vulnerability localization, while parameter and command features provide complementary discrimination for variants that share similar request paths. The regression-based setting achieves the highest F1-score, confirming that the correlation mechanism is more robust than single-feature matching or manually equalized weights.

7. Conclusions

To address the security monitoring requirements in distributed IoT networks and systems, this paper proposes an in the wild vulnerability attack detection method that combines hybrid deep learning classification and open source intelligence correlation. This method alleviates the extreme class imbalance in real traffic through packet preprocessing, uses a BERT CNN hybrid model to automatically identify suspicious attack packets, and achieves precise localization of 1Day/NDay vulnerability types via attack vector regression. On the test set, the proposed method achieves a detection accuracy of 99.99% and an F1 score of 99.98%. The IoT_Exploits_Founder system implemented based on this method has run continuously in a real world environment for one month and discovered 13 new types of in the wild vulnerability attacks, demonstrating the effectiveness and application potential of the method in practical distributed IoT security monitoring scenarios. Future work will be carried out in three aspects: first, to further improve the model's generalization ability for unknown variants and potential 0Day attacks; second, to supplement the evaluation of latency, throughput and resource overhead for online deployment, so as to verify the scalability of the method in larger scale distributed systems; third, to explore the integration with more diverse threat intelligence sources and privacy preserving analysis mechanisms, to better support data security monitoring and continuous operation in distributed networks and systems. The revised experiments further show that the proposed method outperforms representative IoT intrusion and malicious traffic detection baselines, supports batch online inference with 2.11 ms average latency per suspicious packet, and maintains robust open-source intelligence correlation when the threshold and regression weights are varied.

Author Contributions.

Conceptualization, X.W. and X.L.; methodology, X.W. and M.Z.; software, X.Z.; validation, M.Z., X.Z. and K.J.; writing—original draft preparation, X.W. and M.Z.; writing—review and editing, X.L. and K.J.; visualization, M.Z.; funding acquisition, X.L. All authors have read and agreed to the published version of the manuscript.

Funding.
This research was supported by the (No. JC2024036).

References

[1] Neshenko N, Bou-Harb E, Crichigno J, Kaddoum G, Ghani N. Demystifying IoT security: An exhaustive survey on IoT vulnerabilities and a first empirical look on Internet-scale IoT exploitations. IEEE Commun. Surv. Tutorials. 2019;21(3):2702-33.

[2] Sabottke C, Suciu O, Dumitraş T. Vulnerability disclosure in the age of social media: Exploiting twitter for predicting Real-World exploits. In: Proceedings of the 24th USENIX Security Symposium; 2015 Aug 12-14; Washington, DC, USA. Berkeley: USENIX Association; 2015. p. 1041-1056.

[3] Bilge L, Dumitraş T. Before we knew it: an empirical study of zero-day attacks in the real world. In: Proceedings of the 2012 ACM conference on Computer and communications security; 2012 Oct 16-18; Raleigh, NC, USA. New York: ACM; 2012. p. 833-844.

[4] Elder S, Rahman MR, Fringer G, Kapoor K, Williams L. A survey on software vulnerability exploitability assessment. ACM Comput. Surv. 2024;56(8):1-41.

[5] Roesch M. Snort: Lightweight intrusion detection for networks. In: Proceedings of the 13th USENIX Conference on System Administration (LISA); 1999 Nov 7-12; Seattle, WA, USA. Berkeley: USENIX Association; 1999. p. 229-238.

[6] Sommestad T, Holm H, Steinvall D. Variables influencing the effectiveness of signature-based network intrusion detection systems. Inf. Secur. J. 2022;31(6):711-728.

[7] Tavallaee M, Bagheri E, Lu W, Ghorbani AA. A detailed analysis of the KDD CUP 99 data set. In: Proceedings of the 2009 IEEE symposium on computational intelligence for security and defense applications; 2009 Jul 8-10; Ottawa, Canada. IEEE; 2009. p. 1-6.

[8] Shiravi A, Shiravi H, Tavallaee M, Ghorbani AA. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. Comput. Secur. 2012;31(3):357-374.

[9] Ring M, Wunderlich S, Grüdl D, Landes D, Hotho A. Flow-based benchmark data sets for intrusion detection. In: Proceedings of the 16th European Conference on Cyber Warfare and Security (ECCWS); 2017 Jun 29; South Oxfordshire, UK. South Oxfordshire, UK: ACPI; 2017. p. 361-369.

[10] Lee W, Stolfo S. Data mining approaches for intrusion detection. In: Proceedings of the 7th USENIX Security Symposium; 1998 Jan 26-29; San Antonio, TX, USA. Berkeley: USENIX Association; 1998. p. 79-94.

[11] Khan L, Awad M, Thuraisingham B. A new intrusion detection system using support vector machines and hierarchical clustering. VLDB J. 2007;16(4):507-521.

[12] Nguyen TT, Armitage G. A survey of techniques for internet traffic classification using machine learning. IEEE Commun. Surv. Tutorials. 2008;10(4):56-76.

[13] Sommer R, Paxson V. Outside the closed world: On using machine learning for network intrusion detection. In: Proceedings of the 2010 IEEE symposium on security and privacy; 2010 May 16-19; Oakland, CA, USA. IEEE Computer Society; 2010. p. 305-316.

[14] Ring M, Wunderlich S, Scheuring D, Landes D, Hotho A. A survey of network-based intrusion detection data sets. Comput. Secur. 2019;86:147-167.

[15] Ma J, Saul LK, Savage S, Voelker GM. Identifying suspicious URLs: an application of large-scale online learning. In: Proceedings of the 26th annual international conference on machine learning; 2009 Jun 14-18; Montreal, Canada. ACM; 2009. p. 681-688.

[16] Ma J, Saul LK, Savage S, Voelker GM. Beyond blacklists: learning to detect malicious web sites from suspicious URLs. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining; 2009 Jun 28-Jul 1; Paris, France. New York: ACM; 2009. p. 1245-1254.

[17] Zhao P, Hoi SC. Cost-sensitive online active learning with application to malicious URL detection. In: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining; 2013 Aug 11-14; Chicago, IL, USA. New York: ACM; 2013. p. 919-927.

[18] Yun X, Xie J, Li S, Zhang Y, Sun P. Detecting unknown HTTP-based malicious communication behavior via generated adversarial flows and hierarchical traffic features. Comput. Secur. 2022;121:102834.

[19] Hodo E, Bellekens X, Hamilton A, Dubouilh PL, Iorkyase E, Tachtatzis C, Atkinson R. Threat analysis of IoT networks using artificial neural network intrusion detection system. In: Proceedings of the 2016 International symposium on networks, computers and communications (ISNCC); 2016 May 11-13; Yasmine Hammamet, Tunisia. IEEE; 2016. p. 1-6.

[20] Thamilarasu G, Chawla S. Towards deep-learning-driven intrusion detection for the internet of things. Sensors. 2019;19(9):1977.

[21] Muna AH, Moustafa N, Sitnikova E. Identification of malicious activities in industrial internet of things based on deep learning models. J. Inf. Secur. Appl. 2018;41:1-11.

[22] Abdel-Basset M, Hawash H, Chakrabortty RK, Ryan MJ. Semi-supervised spatiotemporal deep learning for intrusions detection in IoT networks. IEEE Internet Things J. 2021;8(15):12251-12265.

[23] Tsimenidis S, Lagkas T, Rantos K. Deep learning in IoT intrusion detection. J. Netw. Syst. Manag. 2022;30(1):8.

[24] Suciu O, Nelson C, Lyu Z, Bao T, Dumitraș T. Expected exploitability: Predicting the development of functional vulnerability exploits. In: Proceedings of the 31st USENIX Security Symposium (USENIX Security 22); 2022 Aug 10-12; Boston, MA, USA. Berkeley: USENIX Association; 2022. p. 377-394.

[25] Adewopo V, Gonen B, Adewopo F. Exploring open source information for cyber threat intelligence. In: Proceedings of the 2020 IEEE International Conference on Big Data (Big Data); 2020 Dec 10-13; Atlanta, GA, USA. IEEE; 2020. p. 2232-2241.

[26] Minaee S, Kalchbrenner N, Cambria E, Nikzad N, Chenaghlu M, Gao J. Deep learning-based text classification: a comprehensive review. ACM Comput. Surv. 2021;54(3):1-40.

[27] Kim Y. Convolutional neural networks for sentence classification. In: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP); 2014 Oct 25-29; Doha, Qatar. Doha: Association for Computational Linguistics; 2014. p. 1746-1751.

[28] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I. Attention is all you need. Adv. Neural Inf. Process. Syst. 2017;30.

[29] Devlin J, Chang MW, Lee K, Toutanova K. Bert: Pre-training of deep bidirectional transformers for language understanding. In: Burstein J, Doran C, Solorio T, editors. Proceedings of the 2019 conference of the North American

chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers); 2019 Jun 2-7; Minneapolis, MN, USA. Minneapolis: Association for Computational Linguistics; 2019. p. 4171-4186.

[30] Saeed S, Suayyid SA, Al-Ghamdi MS, Al-Muhaisen H, Almuhaideb AM. A systematic literature review on cyber threat intelligence for organizational cybersecurity resilience. Sensors. 2023;23(16):7273.