Study on AI-Powered Cybersecurity for Threat Detection and Mitigation

Thein, Thin Tharaphe

(Degree)
博士（工学）

(Date of Degree)
2024-03-25

(Date of Publication)
2025-03-01

(Resource Type)
doctoral thesis

(Report Number)
甲第8933号

(URL)
https://hdl.handle.net/20.500.14094/0100490158

Doctoral Dissertation

Study on AI-Powered Cybersecurity for Threat Detection and Mitigation

(脅威の検知と緩和のためのAIを活用したサイバーセキュリティに関する研究)

January 2024

Graduate School of Engineering,
Kobe University

THIN THARAPHE THEIN

Acknowledgements

First and foremost, I would like to extend my heartfelt gratitude to all those who have contributed to the completion of this thesis. Especially, I would like to express my sincere gratitude to Prof. Yoshiaki Shiraishi for his guidance as academic advisor and huge support in the progression of this research. I am also deeply thankful for the mentorship provided by Prof. Masakatu Morii. Furthermore, I am grateful to all individuals whom have engaged in constructive discussions and provided feedback on my research during various research meetings and symposiums, which have significantly contributed to the development of this research. I would like to give special thanks to Ministry of Education, Culture, Sports, Science and Technology (MEXT) of Japan for providing me with a scholarship that enabled my study at Kobe University. Last but not least, I greatly appreciate my family and friends for their continuous encouragement and unwavering support throughout my academic journey.

Abstract

In today's digital world, cyber security has become a significant concern as cyber threats progressively evolve and become more complex. The ability to quickly detect, respond to, and recover from emerging cyber-attacks has become a critical issue in cybersecurity operations. The dynamic and stealthy nature of cyber-attacks often makes the static defense approach ineffective. With the emergence of AI, AI can serve as a powerful technique for defending against cyber-attacks. By automating repetitive analysis processes with massive amounts of data, AI can generate intelligent data insights and predict outcomes with high precision. Many research and development processes have been dedicated to AI-driven cybersecurity to detect and mitigate large-scale cyber-attacks. However, with the ever-evolving and constantly changing nature of threats, it is essential to regularly update and retrain the AI model to keep up with the changing threat landscape.

Various security vendors and expert communities usually publish the newly emerged threat information in the form of incident reports on the Internet. However, those reports are often written in unstructured text (various formats), and they are not in the machine-readable format to be automatically analyzed and utilized as threat intelligence. Therefore, in Chapter 2 of the dissertation, we introduce the threat modeling techniques with the neural network model to identify the cyber kill chain stages of the unstructured security text. Security experts can utilize the seven stages of the cyber kill chain framework to understand the adversary's specific goals and actions, and they can implement necessary security controls to defend the organization's network.

The objective of this study is to develop a machine learning model that can automatically identify cyber kill chain stages and extract the Indicator of Compromises (IoCs) (e.g., domain names, IP, URL) from unstructured security text, which contains a lot of information about specific security breaches. Moreover, we combine the cyber kill chain framework, which emphasizes the steps involved in a cyberattack, and MITRE ATT&CK, which details the techniques and tactics employed by the attackers, to provide the integrated analysis of threat information. We utilize the example techniques from ATT&CK for Enterprise (each technique is manually labeled with a specific cyber kill chain stage) as a training dataset, which is a novel approach since the training data and test data are from different data contexts (i.e., the training dataset contains techniques of ATT&CK and test dataset is security articles). The technique in ATT&CK represents the adversary's tactical goal and how it is achieved by performing a specific action. The knowledge extracted from our model can be used to build threat taxonomy,

or the derived IoCs can be utilized as a cyber threat intelligence feed. Though this study presents a novel way to identify the unstructured threat information by combining ATT&CK and cyber kill chain framework, the experimental results highlight the need for further research and development to improve the model efficiency.

To further enrich the threat feeds, we also explored the detection of malicious domains by leveraging the domain name's semantic features, lexical, and DNS-based features in Chapter 3 of the dissertation. Our objective is to examine the utilization of different groups of features in order to enhance the detection ability of the malicious domain. First, we introduce the analysis of the Domain Name System (DNS) to find the footprints left by the malicious domain. We employ active DNS to query the DNS-related data for the given domain name and subsequently build the domains' DNS information feature set. The DNS features include domain address records, name server records, mail exchange records, time-to-live, active time, and lifetime of the domain. Additionally, we make a second group of features called lexical features, which comprises the number of consecutive characters, digits, words, and domain length. Furthermore, we introduce semantic features by computing the domain reputation score by the N-gram method.

In the experiment, we examine the effectiveness of each feature group for detecting malicious domains. The experimental results indicate incorporating all groups of features enhances the recognition of the malicious domains. One significant limitation of this study is the collection of the active DNS data since it takes time to query the DNS-related information for every domain in the dataset. Moreover, due to the constraint of the DNS data collection, the dataset used in the experiment is relatively small, which highlights further research and improvements to ensure the robustness of the model.

While threat feeds can be utilized to protect the network from cyber-attacks, additional measures are necessary to defend against newly emerging malware. Therefore, we extend our study scope to network logs analysis to better understand threats and implement robust countermeasures. The research findings of the network logs analysis are described in Chapters 4 and 5 of this dissertation.

We introduced a few-shot graph neural network model to identify malicious IoT network traffic in Chapter 4. The existing malicious traffic detection systems that rely on supervised machine learning require a considerable amount of malware and benign traffic samples to train the effective learning model. Therefore, this study aims to identify the new type of malware traffic with a few labeled traffic samples, thereby minimizing the requirement of model retraining and labeling costs. Given that the primary goal of few-shot learning is to recognize new, unobserved data categories with limited labeled data samples without requiring model retraining for the newly emerged data categories, it aligns well with our objective.

Since the input of our model is an image, we initially preprocess the bidirectional network flow as an RGB color image. We also employed the CNN model pre-trained on ImageNet to extract network features from the transformed network images. Our few-shot model learns how close the two network flows are in the embedding space through training on various few-shot tasks and subsequently generalizing this knowledge to recognize the new unseen attack category. The evaluation results indicate that our model can recognize newly emerged network traffic categories with limited labeled samples at the few-shot testing stage and outperform the baseline models. While few-shot learning offers the advantage that the training and test data categories do not exactly need to be the same, it has limitations. Since the number of classes is fixed during the meta-training stage, if we train the model with N classes, only N classes can be predicted at a time in the meta-testing stage. Moreover, the current methods have limitations in the deployment of the real-time network as it has time complexity and cost associated with internal graph construction. This highlights the need to develop a lightweight and efficient model suitable for real-time deployment.

In Chapter 5, we further extend our network traffic identification model to the privacy-preserving and distributed learning framework, where we can collaboratively train the model and share the threat intelligence without privacy concerns. This time, our model emphasizes tackling the challenges posed by applying federated learning to cybersecurity data, including data heterogeneity and client poisoning attacks. We have analyzed data poisoning and model poisoning attacks with different data partition scenarios (to simulate data heterogeneity) to examine their effects and proposed a robust modeling technique accordingly. The experimental results verify that our model outperforms almost all baseline methods. On a separate note, our model is designed for cross-silo federated learning where the client number is relatively small, and each client has sufficient computing power. In the experiment, we allow every client to participate in every communication round. However, for large-scale IoT devices, this full participation assumption is infeasible as the server needs to wait for all clients. The server must wait indefinitely if one client struggles to complete the model training. This highlights the need for further research and development on client selection and handling of the straggling clients for the federated learning paradigm.

This thesis provides various modeling techniques for cybersecurity datasets, such as deep neural networks, ensemble machine learning, graph neural networks, and federated learning. We have performed comprehensive experiments to demonstrate the effectiveness of the proposed models. We hope our methodologies, insights, and research findings will serve as a foundation in advancing AI-driven cybersecurity.

Contents

Acknowledgements i

Abstract ii

List of Figures viii

List of Tables ix

1 Introduction 1
- 1.1 Background and Motivation . . . 1
- 1.2 AI Techniques in Cybersecurity Domain . . . 3
  - 1.2.1 Machine Learning and Deep Learning . . . 3
  - 1.2.2 Cybersecurity Dataset . . . 4
  - 1.2.3 Performance Evaluation Criteria . . . 4
- 1.3 Data-driven Cyber Intelligence . . . 5
- 1.4 Cyber Threat Intelligence Feed . . . 6
- 1.5 Our Study . . . 7
  - 1.5.1 Paragraph-based Estimation of Cyber Kill Chain Phase from Threat Intelligence Reports . . . 8
  - 1.5.2 Malicious Domain Detection Based on Decision Tree . . . 8
  - 1.5.3 Few-Shot Learning-Based Malicious IoT Traffic Detection with Prototypical Graph Neural Networks . . . 9
  - 1.5.4 Personalized Federated Learning-based Intrusion Detection System: Poisoning Attack and Defense . . . 9
- 1.6 Chapter Organizations . . . 9

2 Paragraph-based Estimation of Cyber Kill Chain Phase from Threat Intelligence Reports 11
- 2.1 Abstract . . . 11
- 2.2 Introduction . . . 11
- 2.3 Background . . . 12
  - 2.3.1 Cyber Kill Chain . . . 12
  - 2.3.2 Diamond Model of Intrusion Analysis . . . 14
  - 2.3.3 Existing Study on Threat Modeling and Threat Extraction . . . 14
- 2.4 Proposed Model . . . 15
  - 2.4.1 Word Embedding . . . 16

2.4.2 Paragraph-based Estimation of the Cyber Kill Chain Phase 16
2.4.3 Core Features Extraction from Paragraph mainly with ATT&CK 18
2.5 Evaluation 18
2.5.1 Results 19
2.6 Conclusion 19

3 Malicious Domain Detection Based on Decision Tree 21
3.1 Abstract 21
3.2 Introduction 21
3.3 Background 23
3.3.1 Existing Study on Malicious Domain Detection 23
3.3.2 Domain Name System 23
3.3.3 DNS Traffic Analysis 25
3.4 Proposed Model 26
3.4.1 Data Collector 27
3.4.2 Feature Extraction 27
3.5 Evaluation 29
3.5.1 Results 30
3.6 Conclusion 31

4 Few-Shot Learning-Based Malicious IoT Traffic Detection with Prototypical Graph Neural Networks 32
4.1 Abstract 32
4.2 Introduction 32
4.3 Background 34
4.3.1 Network Intrusion Detection System 34
4.3.2 Signature-based vs. Anomaly-based IDS 36
4.3.3 Few-shot Learning 36
4.3.4 Graph Neural Networks for Few-shot Learning 37
4.3.5 Existing Study on IoT Network Traffic Detection 38
4.4 Proposed Method 40
4.4.1 Few-shot Learning Strategy 40
4.4.2 Data Preprocessing 41
4.4.3 Few-shot Prototypical Graph Neural Network 42
4.4.4 Training Objectives and Parameters 44
4.5 Evaluation 46
4.5.1 Results 47
4.5.2 Discussion 48
4.6 Conclusion 50

5 Personalized Federated Learning-based Intrusion Detection System: Poisoning Attack and Defense 51
5.1 Abstract 51
5.2 Introduction 51
5.3 Background 54

5.3.1 Federated Learning 54
5.3.2 Existing Study on Federated Learning-based IDS 56
5.3.3 Impact of Statistical Heterogeneity in Client Data 57
5.3.4 Poisoning Attacks against Federated Learning-based IDS 58
5.3.5 Defending Poisoning Attacks in Federated Learning 59
5.4 Proposed Method 61
5.4.1 Convolutional Neural Network 61
5.4.2 Local Training: A Personalized Local Model 62
5.4.3 Server-side Poisoned Client Detector 64
5.5 Evaluation 67
5.5.1 Results 70
5.5.2 Discussion 71
5.6 Conclusion 73

6 Conclusion of Our Study **75**
6.1 Conclusion 75
6.2 Contributions 76
6.3 Future work 78

References **79**

List of Publications **87**

List of Figures

1.1 Overview of data-driven cyber intelligence 6
1.2 Integration of external threat intelligence in SIEM system 8

2.1 Cyber kill chain 13
2.2 Diamond model 14
2.3 Outline of the paragraph-based cyber kill chain model 16
2.4 Proposed cyber kill chain phase classification model 17

3.1 Managing client requests in Domain Name System 25
3.2 Attackers using the Domain Generation Algorithm to compromise computer system 26
3.3 Overview of proposed malicious domain detection model 27

4.1 Network intrusion detection system 35
4.2 Few-shot Learning 37
4.3 Graph Convolutional Neural Networks 38
4.4 Network traffic preprocessing 40
4.5 Visual representation of network flows 42
4.6 Proposed few-shot learning-based network traffic detection system 43
4.7 Confusion matrix of few-shot model on test dataset 49

5.1 Federated learning system 55
5.2 The architecture of the federated learning-baed IDS model 59
5.3 Client drift in FedAvg algorithm 62
5.4 The accuracy and F1 score of pFL-IDS with different ratios of malicious client 72

List of Tables

1.1 Confusion matrix 5

2.1 Kill chain phase classification result 19

3.1 Domain features 28
3.2 Domain reputation scores 29
3.3 Experimental results for malicious domain detection 30

4.1 Color mapping by the Binvis binary data visualization tool 41
4.2 Statistics of image dataset used in experiment 46
4.3 Evaluation results of few-shot learning-based network traffic detection system . 47

5.1 A summary of existing works on federated learning-based IDS 56
5.2 Statistics of the mini-N-BaIoT dataset 64
5.3 Hyperparameters 64
5.4 Evaluation of non-IID data (scenario 1). 69
5.5 Evaluation of non-IID data (scenario 2). 69
5.6 Evaluation of IID data. 70

Chapter 1

Introduction

In this chapter, we introduce the background and motivation of this study, offering the essential concept of utilizing AI to automate the threat analysis process. We also discuss the previous literature which enable to grasp the fundamental concepts that have motivated this research. In addition, we introduce the AI-based methodologies explored in this study and the experimental evaluation of our proposed techniques.

1.1 Background and Motivation

With the advancements in information technology, the Internet has become an indispensable part of our lives. Undoubtedly, all devices connected to the Internet have the potential to be compromised by malware attacks. As the world has become more and more interconnected, it provides multiple interfaces for attackers to conduct cyberattacks. Cybersecurity refers to different techniques, methods, and rules to defend cyberspace against threats, malware, fraud, phishing, and spam. In today's digital world, cybersecurity has become a significant concern as cyber threats have progressively evolved and become more sophisticated. Therefore, the capability to quickly detect, respond to, and recover from ever-growing cyber-attacks has become a critical requirement in cybersecurity operations.

With the emergence of artificial intelligence, adversaries can now incorporate AI techniques to launch targeted cyberattacks [1]. The dynamic and stealthy nature of cyberattacks often makes the traditional defense approaches ineffective. The defense mechanisms that relied only on the security expert's knowledge and signature-based/rule-based strategy have become insufficient to keep up with the growing threat landscape. There is a need for robust countermeasure techniques that can detect both known and unknown threats. However, keeping up with the rapidly expanding cyberspace while delivering an advanced threat detection model is challenging.

While AI is increasingly misused in cyberattacks, it can also serve as a powerful technique for defending against cyberattacks, including AI-powered cyber threats [1]. AI-powered data-driven cyber intelligence framework offers promising solutions to alleviate the ever-evolving cyberattacks. AI can automate data analysis processes and analyze vast amounts of data to identify potential security breaches in real-time. Moreover, AI-enabled data analysis can recognize the threat behavior and patterns that analysts might miss, thereby providing security experts with a deeper insight into the tactics, techniques, and procedures (TTP) utilized by threat

adversaries to make intelligent decisions for threat detection and mitigation. In general, we can categorize the processes involved in data-driven cybersecurity as follows.

- Collection of massive data
- Data analytics with various techniques
- Real-time detection
- Threat mitigation & countermeasures (with expert knowledge)

The most commonly applied AI technique in the cybersecurity domain is machine learning, which offers advantages over traditional detection methods [2]. Machine learning plays a crucial role in combating cyber-attacks. The application areas include phishing detection [3], spam detection [4], malware detection [5], malicious domain detection [6–8], intrusion detection systems [9, 10], and much more. This research primarily focuses on AI-powered cybersecurity, offering threat modeling approaches for unstructured cybersecurity text, malicious domain detection, and intrusion detection systems. We introduce a brief explanation as follows.

- **Unstructured text of cyber threat intelligence (CTI) reports**: CTI is the collection and analysis of vulnerability and threat information that can be utilized to develop various defense mechanisms. CTI can be structured (e.g., STIX-Structured Threat Information eXpression) or unstructured (e.g., blogs, SNS, websites) text. Various security vendors usually publish the newly emerged threat information as incident reports (unstructured text) on the Internet in a timely manner. However, those reports are not in the machine-readable format to be automatically analyzed and utilized as threat intelligence. Research efforts have been dedicated [11, 12] to the automated modeling of unstructured text of CTI sources, which map threat actions to appropriate tactics, techniques, and procedures (TTP) as well as cyber kill chain phases. Nevertheless, there remains potential for improvement.

- **Malicious domain detection**: Domains are fundamental components of the Internet. The adversaries can utilize domain URLs as an attack communications medium and compromise users into victims of phishing or spam. Typical defense approaches include the use of domain blacklists, machine learning, and deep learning techniques. In machine learning techniques, a variety of features such as domain lexical features, domain name system features, domain semantics features, website ranking lists, and much more have been proposed so far. Each approach offers advantages and disadvantages, and examining the best strategies to detect the malicious domain is an ongoing area of interest.

- **Intrusion detection system**: A network intrusion detection system is a protection system that scans network vulnerabilities, monitors computer networks to detect unauthorized intrusions, and eventually takes appropriate defense actions. The fundamental classification approaches to detecting various network intrusions are signature-based, anomaly-based, or a combination of both approaches. Machine learning techniques have also significantly contributed to advancing intrusion detection systems. However, detection of zero-day attacks, data labeling costs, and lack of labeled data are still substantial challenges for machine learning-based intrusion detection systems.

In this study, we investigate the literature on AI-powered modeling techniques and focus on improving the existing AI models in the context of cybersecurity datasets. This study aims to extract valuable threat insights from cybersecurity data sources through AI techniques. Keeping this objective, we propose and develop machine learning/deep learning models for various security data (e.g., security articles, domain names, and network logs). To demonstrate the effectiveness of our proposed approaches, we conducted extensive experiments and a comparative analysis with existing studies.

1.2 AI Techniques in Cybersecurity Domain

Cyber-attacks involve cybercriminals and other adversaries attempting to infiltrate computer systems for personal gain. The victims can range from individual users to enterprises or even government organizations. Until now, we have observed different types of attacks on cyberspace, each imposing varying degrees of damage on the victims. The most common cyber threats and attacks include malware, DDoS, phishing, spam, fraud, malicious code download, and malicious domain/URL. The advancement of Artificial Intelligence (AI) offers a perfect opportunity to detect and mitigate threats in the cybersecurity domain. AI enables the machine to perform tasks that typically require human cognitive abilities. In recent years, AI has become a critical component in cybersecurity as it can analyze millions of data and identify various threats in real-time. Application areas include identifying zero-day vulnerabilities and recognizing malicious behavior such as malware code download, phishing, and network intrusion.

1.2.1 Machine Learning and Deep Learning

Machine learning and deep learning are a sub-field of artificial intelligence that enables the machine to learn hidden patterns and relation in data without being explicitly programmed. The trained machine learning model can make precise predictions and decisions on new and unseen data relevant to the training data. In contrast, the deep learning model is built upon artificial neural networks, consisting of interconnected neurons with multiple hidden layers between the input and output layers, to process the information. Due to the deep layers, deep neural networks often require more training times compared to the machine learning model when training large datasets. Convolutional neural network (CNN) is a variant of deep learning that is designed to process data such as images and is the most commonly used method in computer vision. CNN operates by stacking multiple convolution layers, pooling layers, batch normalization layers, and fully connected layers to learn the spatial relations in data.

Machine learning can be further categorized into supervised, semi-supervised, and unsupervised learning. In supervised learning, the model has knowledge of the targeted label associated with the data; therefore, the model is trained on a fully labeled dataset. Supervised learning facilitates the model to learn the output label associated with input data, subsequently enabling the model to make predictions on unseen data. On the other hand, unsupervised learning is suitable for

training the unlabeled dataset since the model can learn patterns and relations in the data on its own. Semi-supervised learning lies between supervised and unsupervised learning; the model is trained on the dataset consisting of both label and unlabeled data.

1.2.2 Cybersecurity Dataset

Machine learning and deep learning techniques rely on the data to train the learning models. The diversity of the data, the quality of the label, and the amount of data available play a significant role in modeling the AI techniques. Various cybersecurity datasets have been studied in the literature, including private and publicly available datasets. This section introduces some public cybersecurity datasets utilized in the research community.

- **CIC-IDS2017** [13]: This network intrusion detection dataset is obtained by simulating a variety of attacks over a period of five days and contains benign and common cyber attacks, which resemble the actual real-world data. The dataset includes 14 network attacks and benign traffic, such as DoS, DDoS, Botnet, Infiltration, Heartbleed, PortScan, and Web Attack. Moreover, it provides 84 network features retrieved from the analysis with CICFlowMeter. This dataset has been a popular dataset in studying network intrusion detection systems.

- **UNSW-NB15** [14]: This network traffic dataset is created by combining real modern normal activities and synthetic contemporary attack behaviors. The dataset has 49 network features and contains normal traffic and nine types of attacks – Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode, Fuzzers, Analysis, and Worms.

- **N-BaIoT** [15]: This IoT network dataset contains a collection of network traffic from 9 commercial IoT devices infected with Mirai and BASHLITE. The dataset has 115 features and more than 70 million traffic samples.

- **CIC-AndMal2017** [16]: It is an Android malware dataset obtained from executing malware and benign applications on real smartphones. It consists of four malware categories: Adware, Ransomware, Scareware, and SMS malware. Moreover, the dataset provides more than 80 features extracted by CICFlowMeter.

- **ISCX-URL2016** [17]: It provides a lightweight dataset to detect and categorize malicious URLs according to their attack type. It comprises 79 features and 5 URL categories – benign, phishing, malware, defacement, and spam.

1.2.3 Performance Evaluation Criteria

The confusion matrix is a table that is used to measure the performance of the machine learning and deep learning algorithms. Precisely, true positive (TP), true negative (TN), false positive

TABLE 1.1: Confusion matrix

		Predicted Class	
		Benign	Malicious
Actual Class	Benign	TP	FP
	Malicious	FN	TN

(FP), and false negative(FN) are used to compute the confusion matrix as indicated in Table 1.1. FP indicates the number of normal data incorrectly classified as an attack, while FN represents the number of attack samples incorrectly classified as normal. TP and TN denote the number of normal or attack data that were accurately classified. The evaluation metrics, such as precision, recall, F1 score, and accuracy, are computed based on these four values.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}}, \quad \text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \quad \text{F1} = 2 * \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

1.3 Data-driven Cyber Intelligence

The basic principle of forensic science is that every action left a trace, which we can analyze to know what happened exactly. In the area of cybersecurity, a vast amount of data comes from sources such as network logs, Portable Executable (PE) malware, and publicly available threat information from expert communities. We can utilize such data to analyze, detect, and build defense strategies to mitigate potential threats. Since AI can identify trends and hidden patterns in the threat landscape, we can effectively employ AI to create a cyber defense system [18]. Typically, data-driven cybersecurity involves automated data analysis and interpretation to derive valuable knowledge from enormous threat data [19] [20]. Subsequently, these extracted insights can further assist security experts in making countermeasure decisions to protect critical infrastructure from cyber-attacks. Utilizing data-driven intelligence, we can automate cybersecurity tasks such as network logs analysis, malware analysis, malicious domain detection, and network intrusion detection. Subsequently, we can build the cybersecurity taxonomy or knowledge graph to develop a more capable cyber defense system. Fig. 1.1 illustrates the data-driven threat analysis framework, which combines the human expert's knowledge and automated data analytics to provide intelligent cyber solutions.

The following section discusses the processes involved in the data-driven threat intelligence framework for cybersecurity risk management. To extract valuable knowledge from data, gathering comprehensive data, data preprocessing, and effective modeling techniques are some of the most important steps in the data-driven strategy. Collecting comprehensive cybersecurity

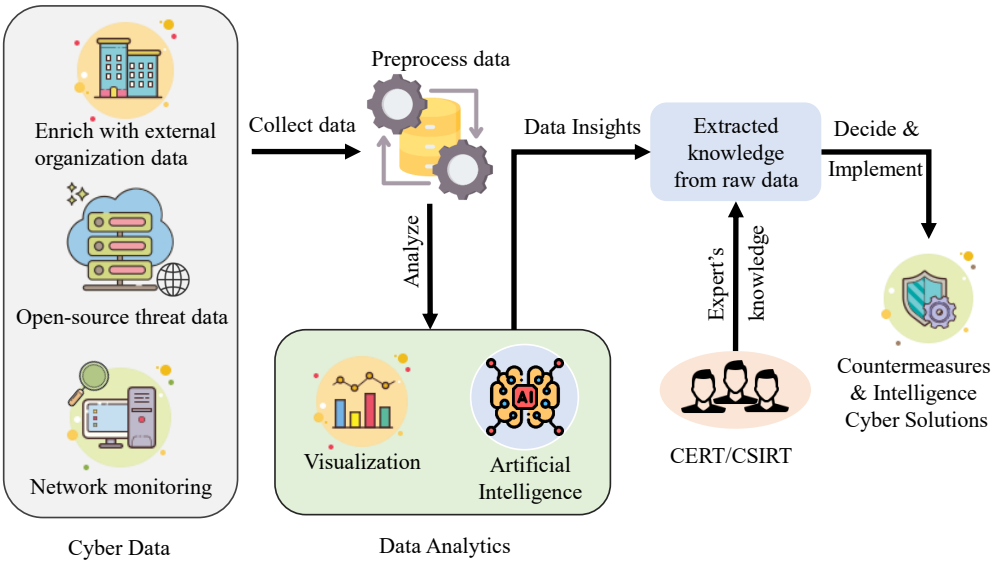

Fig. 1.1. Overview of data-driven cyber intelligence

data is extremely important, as the more data available, the better the AI model can learn. Data can be gathered from network logs, network traffic security reports, PE malware, and system events. We need to clean and preprocess the collected data as required by the AI model. Following this, data can be labeled with expert knowledge; some synthetic data can be generated if data categories are highly imbalanced. Moreover, it is necessary to extract important and meaningful features from the processed data to fit the data into the target learning model. Finally, we can develop a suitable AI model while considering the data's specific characteristics. Various models such as random forest, neural networks, graph neural networks, few-shot learning, and federated learning can be utilized to develop a remarkable learning model. Subsequently, the effectiveness of trained AI models can be evaluated using evaluation metrics such as accuracy, precision, recall, or F1 score. Since the AI models are not 100% accurate, the extracted knowledge supposedly need to be combined with security experts' knowledge to make critical decisions. Data-driven cyber intelligence can be applied in various application areas, such as analyzing massive volumes of data generated by IoT devices and detecting anomalies in critical infrastructure, digital twins, smart cities, industrial control systems, and healthcare.

1.4 Cyber Threat Intelligence Feed

A threat intelligence feed is a series of data comprising knowledge about potential cyberattacks gathered from a variety of sources. The threat intelligence feed can be enriched in various ways, some of which are described as follows.

- Monitoring and analysis of network traffic
- Malware analysis
- Web Crawling to identify attacks such as phishing
- Open-source security-related data

- Insights from security experts
- Adversaries' Tactic, Technique, and Procedure (TTP)

This feed contains threat data such as IoCs, suspicious IPs and domains, and malware signatures and is continuously being updated to include the latest information about the threats. By utilizing this threat feed, it is expected to be able to predict future attacks by assessing interrelated threat actions between existing attacks, enabling security professionals to implement proactive countermeasures. This feed can be used to block known malicious sources, detect network anomalies, and generate alerts if suspicious behaviors are observed. In addition, the threat intelligence feeds can be in a structured format (e.g., STIX-Structured Threat Information expression) or unstructured text (e.g., blog posts, SNS tweets, websites). The structured feeds are machine-readable and can be utilized immediately by SIEM – security information and event management – and other cybersecurity systems to block emerging threats at the earlier stages. On the other hand, an unstructured text format allows human analysts to have a deeper understanding of the new unseen threats since the detailed analysis of new threat information is often published in a timely manner by security experts.

Generally, SIEM is a security solution designed to detect potential threats and is a key component of the security operations center (SOC). SIEM system collects, evaluates, and analyzes the network logs to recognize suspicious threat behaviors before they can compromise the entire network. SIEM tools are great at detecting known attacks; however, the constant evolution of threats imposes a major challenge for SIEM tools in keeping up with the new threat behaviors and techniques employed. In addition, threat information of both structured and unstructured formats should be utilized to comprehensively understand threat behaviors and improve network protection. As illustrated in Fig. 1.2, the integration of SIEM with cyber threat intelligence feed enables the SIEM tools to maximize theirs potential. Since threat feeds are obtained from diverse sources, this integration provides the SIEM with threat insights from a more global perspective. This dissertation focuses on modeling various cybersecurity data to extract valuable insights, identify threat behaviors, and detect threats or suspicious network activities. The research findings from this dissertation can serve as a foundation to create a cyber threat intelligence feed, which, in turn, can be used together with SIEM services to empower the identification of real-time cyber intrusion.

1.5 Our Study

In this study, with an emphasis on advancing AI-powered cybersecurity operations, we proposed various machine learning/deep learning models using cyber data (e.g., adversaries techniques, malware domain, and network traffic logs). Our objective is to automate the data analysis processes and identify the data samples, providing deeper insights into threats with high-level precision. We have employed a variety of learning approaches, such as deep neural networks, ensemble machine learning, graph neural networks, and federated learning. Our research findings are summarized in the following Section.

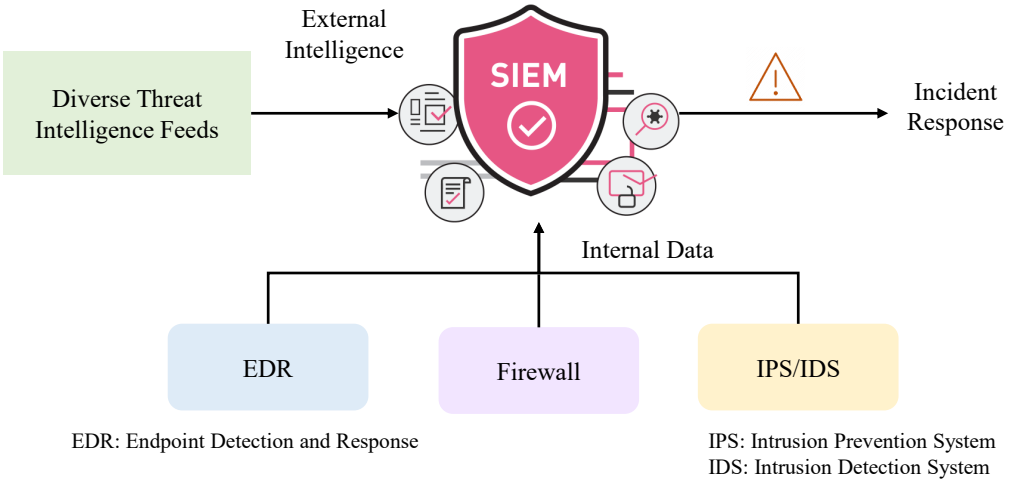

Fig. 1.2. Integration of external threat intelligence in SIEM system

1.5.1 Paragraph-based Estimation of Cyber Kill Chain Phase from Threat Intelligence Reports

In Chapter 2, we designed a neural network model to identify the cyber kill chain phase. The objective is to automate the security text analysis process and to determine which security text paragraph indicates a specific cyber kill chain phase so that we can have a deeper understanding of the techniques utilized by the adversaries. Firstly, our model learns the adversary's techniques available in ATT&CK framework, enabling it to recognize context words related to specific cyber kill chain phases. After that, we evaluated the trained model with the security reports, predicting the kill chain phase for each paragraph of the reports. We employ word2vec to understand semantic similarity among words. While there are 7 cyber kill chain phases, our model is intended to identify the last 5 phases, as the first 2 phases are rarely described in security reports.

1.5.2 Malicious Domain Detection Based on Decision Tree

In Chapter 3, we presented a modeling technique to detect malicious domain names. Given that malicious domains are essential for adversaries to run malicious activities and to compromise user devices, we aim to investigate the effectiveness of leveraging the domain name's semantic features in addition to the most commonly used DNS-based and lexical features. We evaluated our proposal with random forest, XGBoost, and AdaBoost and demonstrated that incorporating the semantic features enhances the recognition of the malicious domains.

1.5.3 Few-Shot Learning-Based Malicious IoT Traffic Detection with Prototypical Graph Neural Networks

In Chapter 4, we designed a model utilizing few-shot learning and graph neural networks to identify malicious IoT network traffic. Few-shot learning aims to recognize new, unobserved data categories with limited labeled data samples without requiring model retraining for the newly emerged data categories. In the data preprocessing process, we transform bidirectional network flows into images to train the few-shot model. After that, the pre-trained CNN model is employed to extract network features. The model learns how close the two network flows are in the embedding space by the proposed prototypical graph neural network. The IoT-23 dataset is used in the experiment to evaluate the performance of the proposed model; the results are compared with the baseline models. We demonstrated that with a few labeled samples at the inference (meta-testing) stage, the graph-based few-shot model can recognize newly emerged network traffic categories (which are not observed during the meta-training stage).

1.5.4 Personalized Federated Learning-based Intrusion Detection System: Poisoning Attack and Defense

In Chapter 5, we introduced a robust federated learning-based intrusion detection system for heterogeneous IoT data. Data heterogeneity and poisoning attacks launched by malicious clients in federated learning are the main focus of this study. We proposed a personalized federated learning approach to tackle the data heterogeneity problem and designed a poisoned client detector at the server to combat the poisoning attacks. We examined two poisoning attacks: data poisoning and model poisoning, and different data partition scenarios: IID (Independently and Identically Distributed) and non-IID. With the extensive experiment conducted, we demonstrated that our model is effective in defending against both poisoning attacks regardless of the data distribution of the clients. Moreover, the empirical results indicated that our approach outperforms almost all baseline methods when experimented with non-IID data and in the presence of poisoning attacks.

1.6 Chapter Organizations

The remaining chapters are organized as follows.

- Chapter 2 introduces the modeling technique for the unstructured text of CTI resources, which are critical in grasping the constantly evolving threat behaviors. The aim is to enrich the cyber threat intelligence feeds with extracted IoCs and threat mapping of adversaries' techniques to cyber kill chain phases, enabling the security professional to better defend the organization's network.

- Chapter 3 aims to enrich the cyber threat intelligence feed with information about malicious domains. This chapter explores the integrated analysis of diverse features to enhance detection accuracy.
- Chapter 4 broadens the research scope to network log analysis. This chapter explores the few-shot learning model for network traffic analysis, thereby, minimizing the requirement of model retraining and labeling costs associated with newly emerged attacks.
- Chapter 5 extends the network analysis model to a privacy-preserving and robust federated learning approach. This chapter addresses the two key issues of the federated learning model, data heterogeneity and poisoning attacks, with experimental results validating that the proposed model effectively handles those issues.
- The final chapter, Chapter 6, provides a comprehensive summary, highlights key contributions, and outlines the potential future research directions to enhance data-driven cyber threat intelligence.

Chapter 2

Paragraph-based Estimation of Cyber Kill Chain Phase from Threat Intelligence Reports

2.1 Abstract

In order to keep up with the increasing number of cyberattacks, defense tactics require a timely and accurate understanding of the threats and corresponding risks. Our proposal introduced an approach for modeling threat information available in unstructured security text to predict a specific cyber kill phase for each paragraph of the security articles. Subsequently, we also extracted the core features of the diamond model from security articles. The experimental results indicated that the model achieved an average F1-score of 0.67, with an average accuracy of 65% in identifying the cyber kill chain phases. Moreover, 86% of the Diamond Model's core features are correctly extracted through pattern matching.

2.2 Introduction

A large number of network attacks, including Advanced Persistent Threat (APT), have been targeting various organizations in recent years. Most APT attacks can evade detection and conduct potentially destructive long-term attack activities using sophisticated intrusion routes. To mitigate such attacks, many security operators, engineers, and researchers have focused on the field of Cyber Threat Intelligence (CTI) for effective threat intelligence sharing. CTI is the collection and analysis of vulnerability and threat information. It can be easily accessible and employed in implementing various proactive defense measures. By utilizing CTI, it is expected that future attacks can be predicted based on existing threat information by assessing interrelated actions between various attacks. It is, therefore, necessary to analyze diverse threat information from various trusted CTI sources.

SIEM is a critical tool in cybersecurity that collects, evaluates, and analyzes the network logs to recognize suspicious threat behaviors. SIEM tools are great at detecting known attacks; however, the constant evolution of threats imposes a major challenge for SIEM tools in keeping up with the new threats. The integration of SIEM with the CTI feeds enables the SIEM tools

to maximize their potential. In general, CTI can be categorized as structured (e.g., STIX-Structured Threat Information eXpression) or unstructured (e.g., blogs, SNS, websites) text. Intending to increase cyber security awareness, various organizations often share analysis of attack information in the form of security reports. In order to use the latest threat information promptly, it is necessary to analyze and utilize unstructured threat data. The current challenge is the lack of an automatic threat modeling procedure.

Therefore, this study proposes a neural network-based automated threat modeling approach to analyze threat information from various security reports. This study assesses security reports at a paragraph level, assuming each paragraph describes one cyber kill chain event. The proposed model maps the cyber kill chain stages to each paragraph of the CTI reports and also extracts IoCs from the reports. The contributions of this study are summarized as follows.

1. We propose an automated modeling of threat information from unstructured text of CTI sources.

2. We combine the cyber kill chain framework, which emphasizes the steps involved in a cyberattack, and MITRE ATT&CK, which details the techniques and tactics employed by the attackers, to provide the integrated analysis of threat information.
3. We utilize the example techniques from ATT&CK for Enterprise as the training dataset, while the testing dataset consists of CTI reports, providing a novel approach to mapping threat information to adversaries' techniques and cyber kill chain phases

4. We utilize the diverse word lists available on the Internet to extract the IoCs from the security reports.

5. We discuss how the knowledge and IoCs extracted from the model can be used to build threat taxonomy, utilizing it as a cyber threat intelligence feed in the field of data-driven cybersecurity.

2.3 Background

2.3.1 Cyber Kill Chain

Cyber Kill Chain [21] is an intelligence-driven framework for analyzing intrusion detection and attack activities. The model has seven distinct stages, as illustrated in Fig. 2.1. These stages enable the security analysts to understand an adversary's tactics, techniques, and procedures and combat APTs, ransomware, and security breaches. To achieve their objectives, the adversary must go through a series of stages (chain) from reconnaissance to actions on objectives. Any disruption at any stage in this chain will interrupt the entire attack process. All cyberattacks – whether phishing, ATP, or ransomware can be mapped to the cyber kill chain activities. Generally, an APT goes through seven phases: Reconnaissance, Weaponization, Delivery, Exploitation, Installation, Command and Control (C2), and Actions on Objectives.

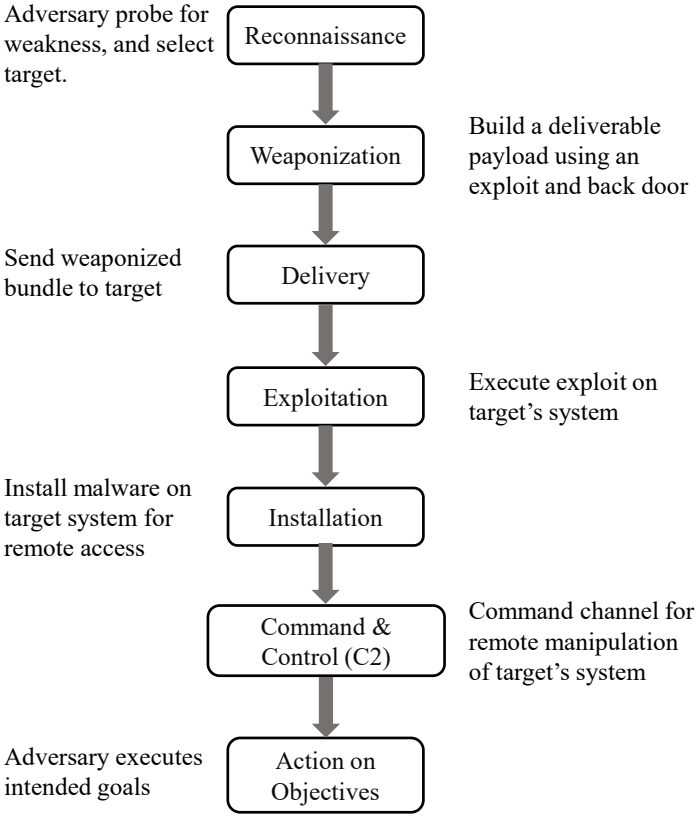

Fig. 2.1. Cyber kill chain

1. **Reconnaissance**: It is the first stage in the cyber kill chain, where adversaries identify potential victims and search for vulnerabilities to find the entry point to the target system.

2. **Weaponization**: This stage takes place after the reconnaissance stage, where the adversaries have obtained information about the potential vulnerabilities. The attackers create a new type of malware or modify the existing malware to compromise the targets.

3. **Delivery**: This stage delivers the cyberweapons mentioned in stage 2 to infiltrate the target system. The weapons can be transmitted via email attachments containing the phishing URLs.

4. **Exploitation**: Once the weapon is successfully delivered, the malicious code starts exploiting the vulnerabilities of the computer system.

5. **Installation**: Cybercriminals install a malware backdoor on the targeted system after exploiting the target's vulnerabilities to gain access to the computer system.

6. **C2**: The attackers communicate with the installed malicious system to carry out their objectives. If the weapon is botnet malware, this malware overloads the C2 servers with thousands of requests to compromise the victim machine as a botnet.

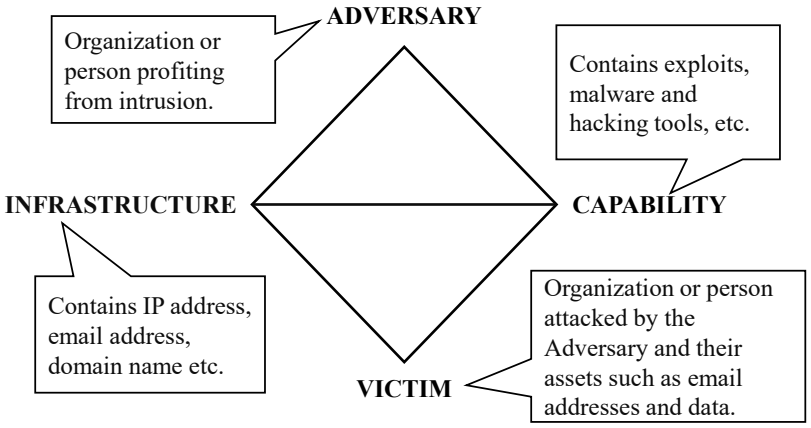

Fig. 2.2. Diamond model

7. **Actions on Objectives**: After successfully completing all of the above stages, the attackers carry out their objectives, which may be distributing malware to steal confidential information or conducting DDoS attacks with botnet malware.

2.3.2 Diamond Model of Intrusion Analysis

The Diamond Model of intrusion analysis [22], typically used alongside the Cyber Kill Chain model, integrates the series of attack activities performed by adversaries. As illustrated in Fig. 2.2, the Diamond Model has four core features: adversary, infrastructure, capability, and victim. These components are referred to as events. Within this model, each event, which is a minimum unit of the chain, is denoted by a diamond shape, with the four events positioned at four vertices of the diamond. The adversary is either an attacker or an organization that leverages its capability (tools and techniques) against the victim to achieve specific goals. The infrastructure, consisting of the logical or physical communication system, is used by the adversary to deliver capability, maintain control, and gain benefits from the victim. Such infrastructure may include e-mail addresses, domain names, and IP addresses. The victim is the primary target of the adversary. In summary, the main axiom of this model states that ”For every intrusion event, there exists an adversary taking a step towards an intended goal by using a capability over infrastructure against a victim to produce a result” [22].

2.3.3 Existing Study on Threat Modeling and Threat Extraction

There has been extensive research on modeling techniques that can automatically extract valuable threat information from unstructured data in media such as online security forums, blogs, and threat reports. Hutchins et al. [21] introduced a method that categorizes APT attacks into various kill chain phases to better understand attacker actions, steps, and motives. Ito et al. [23] proposed a modeling procedure for incident information described in the unstructured CTI

reports. The authors stated that the integrated analysis of the threat information can be obtained by utilizing a relevant relation between CTI reports and threat models.

Husari et al. [12] proposed context-aware modeling to learn attack patterns from the unstructured text of CTI sources to develop the defense mechanism. They utilized Natural Language Processing and information retrieval to extract threat actions based on a semantic relationship and map the extracted threat actions to appropriate tactics, techniques, and a kill chain phase. After that, the authors generates the CTI report formatted according to the STIX (Structured Threat Information eXpression) standard, which is machine-readable.

Zhu et al. [11] proposed ChainSmith, a system that can automatically extract the Indicators of Compromise (IoCs) from security articles and classify them according to their corresponding kill chain phases. The key intuition behind this system is identifying the context words in adjacent sentences in security articles, which denotes a specific kill chain phase. Furthermore, the context words directly related to the IoC determined the level of maliciousness for that IoC. To learn the semantics similarity among words, ChainSmith utilized a dependency-based wording embedding technique [24], employing word dependencies rather than just context words.

This study explores a novel threat modeling approach to identify the cyber kill chain stages of the unstructured text of CTI reports. We use the adversaries' techniques from MITRE ATT&CK for Enterprise as training datasets, while the test dataset is composed of CTI reports from various security vendors. This allows an innovative way of mapping threat actions from security reports to adversaries' techniques, eventually identifying the cyber kill chain stages described in the paragraph level of CTI reports, differentiating this study from previous studies. Moreover, the proposed model extracts IoCs from the CTI reports, which can be utilized as a threat intelligence feed to detect potential threats automatically by combining them with the SIEM tools.

2.4 Proposed Model

The objective of this study is to establish a modeling technique to identify the kill chain phases for threat information described in publicly available unstructured security text. We assumed that the specific threat events are described in each paragraph level of security reports. The proposed model was designed to do two operations: (i) analyze security articles at the paragraph level and predict the corresponding cyber kill phases and (ii) extract core features of the Diamond Model according to the predefined rules. Fig. 2.3 illustrates the two operations of our model. Initially, the model estimated the kill chain phases of the unstructured text and then extracted informative words related to the core features of the diamond model from each paragraph. To further facilitate the core feature extraction process, we have utilized the ATT&CK [25] since it provided comprehensive lists of attackers and malware and other word lists from Wikipedia and the New General Service List (NGSL) [26]. These word lists are employed to extract the Diamond Model's core features through pattern matching.

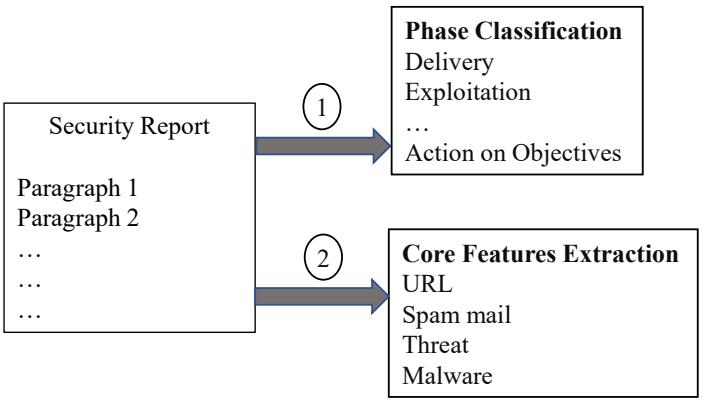

Fig. 2.3. Outline of the paragraph-based cyber kill chain model

2.4.1 Word Embedding

In order to find the semantic similarity among words, the state-of-the-art word2vec [27] algorithm is used in the proposed method to parse words semantically. The word2vec takes the text corpus as input and generates the numerical representations of word features. The word vectors generated by word2vec occupy much less space than one-hot encoded vector. Moreover, word2vec preserves the word's semantics by grouping similar words within a shared vector space.

2.4.2 Paragraph-based Estimation of the Cyber Kill Chain Phase

Since the security report hardly describes the Reconnaissance and Weaponization phase of the cyber kill chain framework, the remaining five phases, Delivery, Exploitation, Installation, Command and Control, and Action on Objectives, are the kill chain phases identified in this study. Fig. 2.4 illustrates the procedures for estimating the cyber kill chain phases. Our model employed five binary neural network classifiers, constructed in the same way described in [11]. Zhu et al. [11] utilized multiple binary classifiers to predict which kill chain phases each sentence represented. The authors aimed to extract IoCs from each sentence as much as possible to construct the structured threat information.

Contrary to the previous approach, our study focused on labeling the cyber kill chain phases for unstructured security articles at the paragraph level and deriving the Diamond event information. Therefore, instead of predicting the kill chain phases of each sentence as indicated in [11], our model operates at the text paragraph level. The proposed kill chain classification model is trained by the example sentences from ATT&CK for Enterprise, a knowledge base managed by MITRE corporation, which categorizes the attacker's behavior in terms of Technique and Tactics. After that, the trained neural network is employed to predict the kill chain phases of the security reports at the paragraph level.

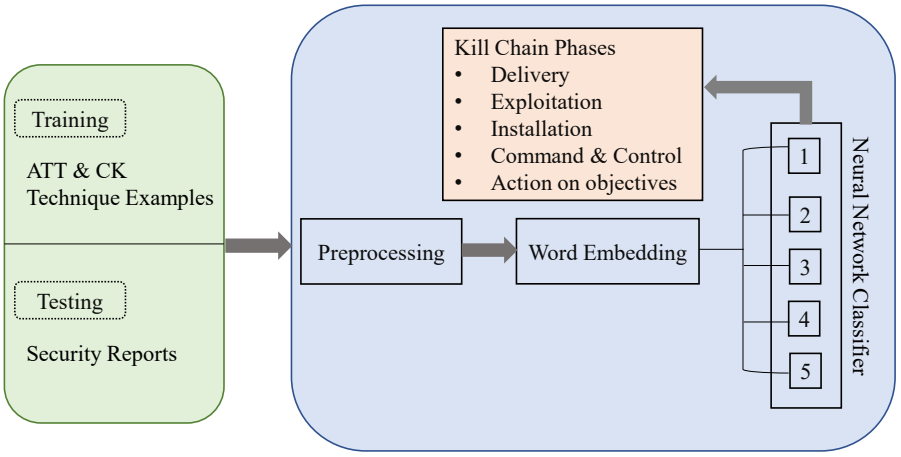

Fig. 2.4. Proposed cyber kill chain phase classification model

Before the model training, the input text corpus is preprocessed with off-the-shelf Natural Language Processing (NLP) techniques. This step performs lowercase conversion, removal of stop-words, punctuation, and special characters. Next, each sentence is tokenized into words, and lemmatization is applied to each word. After this process, each word is parsed using word2vec, which puts semantically similar words in a close position in the vector space. The word vector is trained with the embedding dimension of 100. In the next step, the kill chain phases are estimated by five binary classifiers. The classifier is designed with input, output, and one hidden layer with 50 nodes. This step identifies the features to be fed into the classifier. Firstly, informative words are computed by using the following equation:

$$Score(w) = \max_{k \varepsilon K} \frac{p(w|k)}{p(w)}, \tag{2.1}$$

where the word w represents one of the words appearing in all documents, $p(w)$ represents the probability of occurrence of the word w, k represents one of the set K of all kill chain phases, and $p(w|k)$ is the probability of occurrence of the word w in all documents describing k. These probabilities are used to calculate a score $Score(w)$, which represents the degree to which the word w is specific to a particular kill chain phase. We consider informative words to be words with a high score and a high occurrence. Following this, the context words for each sentence that will be put into the classifiers are calculated. The context words are derived from two statements: (i) informative words of the current sentence and (ii) informative words of previous sentences if no informative words are found in current sentences. The average word embedding of the context words, the words used in the title, and the number of IoCs in each paragraph are then passed into the neural networks as input features. Finally, the classifiers are trained to determine whether each paragraph unit of the security report corresponds to any of the five kill chain phases.

2.4.3 Core Features Extraction from Paragraph mainly with ATT&CK

The objective of the proposed model is to extract core feature words of the Diamond Model, without categorizing them into four types: Adversary, Infrastructure, Capability, and Victim. Since the keywords related to victims are rarely described in the security text, our model extracts only three categories of core features: Adversary, Infrastructure, and Capability. Moreover, words that do not fit into these three categories but can be regarded as potential core features are also retrieved. We put those words into the categories of Candidate core feature words. As a result, the proposed model extracts four categories of core features without specifically categorizing them. Since the core feature words are retrived from the security text through predefined rules, the word lists for pattern matching are generated based on the following statements.

1. Computer related words described in Wikipedia [28]
2. Software names such as malware and tools, etc. described in ATT&CK
3. Group names of attack activities described in ATT&CK
4. New General Service List (NGSL) [26]

The IP address, URL, e-mail address, file name, and CVE are retrieved using the IoCs extraction tool called Cyobstract []. The extracted IP addresses, URLs, and e-mail addresses are categorized as Infrastructure, while file names and CVEs are grouped under Capability. Next, we check the remaining words to determine whether the words match the words lists stated by 1, 2, and 3. If the words are matched, they are extracted and assigned to the categories of Candidate Words, Capability, and Adversary, respectively. After that, we checked if the remaining words matched the NGSL word list.

2.5 Evaluation

We evaluated the effectiveness of the proposed model with the manually labeled dataset since no labeled datasets are available. This study aims to classify the cyber kill chain phase of the security reports as correctly as possible, leveraging the knowledge acquired from the adversary techniques from the ATT&CK framework.

Dataset: Since there were no publicly available labeled datasets for identifying the cyber kill chain phases, we collected data from ATT&CK for Enterprise, Trend Micro [12], and McAfee [13] and used it as the training and test dataset. We employed ATT&CK as a training dataset since the adversary tactics and techniques of ATT&CK are correlated to the cyber kill chain phases. We manually labeled a total of 3101 example sentences of adversary techniques to their corresponding kill chain phases. Additionally, we gathered and tagged labels for four security reports published by Trend Micro and McAfee between November 2018 and December 2018. We assigned a cyber kill chain phase to each paragraph of security reports. This labeled dataset is used as a ground truth in evaluating the effectiveness of the proposed model.

TABLE 2.1: Kill chain phase classification result

Kill chain phase	Accuracy	F1-score
Delivery	0.63	0.72
Exploitation	0.70	0.80
Installation	0.62	0.70
Command & Control (C2)	0.51	0.45
Action on Objectives	0.79	0.70
Average	0.65	0.67

2.5.1 Results

We trained our model on the manually labeled ATT&CK dataset and evaluated the gathered unstructured security articles. The predicted cyber kill chain phases and extracted core features of the Diamond Model are compared with the manually annotated ground truth data. Our evaluation is based on the following two statements:

1. Can the model correctly identify the cyber kill chain phase associated with each paragraph of the security report, evaluated in terms of accuracy and F1-score?

2. Can the model effectively extract the core features of the Diamond Model using recall metrics? Our goal is to retrieve the core features as much as possible without categorizing them into the four types.

The experimental results indicate that the proposed model achieves an average accuracy of 65% and an F1-score of 0.67, as shown in Table 2.1. Out of five kill chain phases, while the accuracy of C2 phase is relatively low, the remaining four phases perform pretty well; notably, the action on objectives phase achieves approximately 80% accuracy. Furthermore, the model successfully extracts the core features from the security reports with 86% recall.

While the proposed model has fulfilled its objective of implementing an automated modeling procedure for threat information within the CTI reports, the experimental results highlight the need for further improvement. One contributing factor to these results is the relatively small size of the training dataset, which contains only 3101 short example sentences of the adversary techniques (TTP). It is possible that the proposed model may not have been able to learn the underlying semantics in the training data. Despite this, our study provides insights into the innovative way of mapping adversaries' techniques to cyber kill chain stages and utilizing this data as the training dataset. It's worth highlighting that the testing dataset contains CTI reports, which have different text structures compared to the training data but still hold relevant information.

2.6 Conclusion

We introduced a method for estimating the cyber kill chain phases and extracting core features of the Diamond Model from the paragraph-level analysis of the security reports. The classification

model trained with ATT&CK in experiments, estimated the cyber kill chain phases with an average F1-score of 0.67 and an average accuracy of 65%. Moreover, the model successfully extracts the core features from the security reports with 86% recall. In future work, we can also evaluate the current model on a dataset consisting of several technical reports garthered from various reputable security blogs or forums. This study can be further extended to structure the extracted threat information into STIX format to share our acquired threat knowledge through the CTI feeds.

Chapter 3

Malicious Domain Detection Based on Decision Tree

3.1 Abstract

Different types of malicious attacks have been increasing simultaneously and have become a serious issue for cybersecurity. Malicious domains play a critical role in engaging in malicious activities over the Internet. The attackers leverage domain URLs as an attack communications medium and compromise users into victims of phishing or spam. While existing domain and IP blacklists provide a way to block malicious domains, these blacklists cannot keep up with the continual increase in newly registered domains. Furthermore, attackers employ a more sophisticated method to evade domain blacklisting attempts, raising the difficulty level of the malicious domain detection algorithms. To address these issues, we propose a machine learning-based detection technique that automatically identifies a domain's maliciousness. Our approach incorporates DNS-based, lexical, and semantic features to enhance the detection capability. The experimental results indicated the effectiveness of our proposal, highlighting the model's capability to detect malicious domains with an approximate accuracy of 0.927 with the random forest classifier.

3.2 Introduction

With the advancements in information technology, the risk and complexity of cybersecurity threats are increasing at an alarming rate, and various malicious cyber-attacks emerge daily. Undoubtedly, all computers connected to the Internet have the potential to be compromised by malware attacks. As early as 2007, security specialists believed that nearly 16-25% of the devices connected to the Internet were part of the botnets [29]. In general, malicious domains are vital for attackers to run malicious activities over the Internet and infect user devices. Attackers can compromise users to be the victims of spam, phishing, and drive-by-download. Subsequently, the attackers can compromise user privacy, install malware, or cause financial losses. Therefore, it is critical to discover and block such malicious activities.

Although the existing domain and IP blacklists can be used to block malicious domains, these blacklists cannot keep up with the continual increase in newly registered domains. Moreover, the attacker often utilizes command and control (C&C) servers to control the compromised hosts.

In earlier days, to establish the connection between the compromised hosts and the C&C servers, attackers used hard-coded domain names or IP addresses of the C&C servers in malware binary. Since those C&C servers with hard-coded IP or domain names can be reverse-engineered and blocked easily by malware analysts, more sophisticated approaches are being employed by the attackers to evade blacklisting attempts.

To avoid C&C servers from being discovered, the attacker frequently changes the domain names for the C&C servers through the Domain-Flux technique. Domain-Flux employed the Domain Generation Algorithm (DGA), which generates a large number of pseudo-random domain names for the C&C server by a random seed. Out of these algorithmically generated domains, the attacker only needs to register a small subset of the domain to C&C servers, where each domain is associated with one or a few IP addresses. As a result, blocking these constantly changing domain names becomes a hassle since these domains are short-lived.

Therefore, an effective approach for accurate and timely detection of malicious domains is crucial in cybersecurity. The Domain Name System (DNS) is a remarkable resource for malicious domain detection, and various detection techniques have been proposed in the research community, which include the analysis of domains using DNS data, statistical features, and the semantic relationship of the domain names. Moreover, the detection model can be featured-based machine learning [6–8], which relies on human-engineered features extracted from domain name strings, or featureless deep neural networks [30, 31], which generate features as part of the training algorithm.

However, the previous studies [32–34] demonstrated that the domain detection algorithm, which relied only on the analysis of the domain name string, can be easily manipulated to evade the detection algorithm. Some researchers outlined that it is challenging for attackers to manipulate the typical domain relationships in DNS traffic. Their observation suggested that if the domain with unknown maliciousness has a strong association with the known malicious domains, then that unknown domain is most likely to be malicious. Following this intuition, we propose the malicious domain detection technique by incorporating the DNS-based, lexical, and semantic features. Our classification model utilizes random forest, XGBoost, and AdaBoost to estimate the maliciousness of a given domain. The contributions of our research are as follows.

1. We propose the malicious domain detection model by incorporating different groups of features to enhance the detection capabilities.

2. We introduce the analysis of DNS traffic to examine the footprints left by the normal and malicious domains. We employ active DNS to query the DNS-related data for the given domain name.

3. We extract DNS features, lexical features, and semantic features from the domain names. The DNS features comprise features such as domain address records, name server records, mail exchange records, time-to-live, active time, and lifetime of the domain, while lexical

features comprise the number of consecutive characters, digits, words, and domain length. The semantic feature is obtained from the domain reputation score computed by the N-gram method.

4. Our experiment results indicate that incorporating all feature groups effectively enhances the recognition of malicious domains.

3.3 Background

3.3.1 Existing Study on Malicious Domain Detection

The detection methods for malicious domains proposed in previous studies can be categorized into classification-based and graph-based approaches.

1. **Classification-based Approach**: This approach mainly employs machine learning algorithms with manually extracted features from domain names and DNS traffic data. Examples of such features, which can differentiate the legitimate and malicious domains, are the domain length, number of characters, number of digits, and time-to-live (TTL). Bilge et al. [6] proposed a system known as EXPOSURE, which can detect malicious domain names by using a decision tree algorithm with features extracted from passive DNS analysis. Similarly, Messabi et al. [7] introduced a decision tree-based malware detection method that relies on DNS records and domain name features to identify malicious domains. Chiba et al. [8] proposed the DomainProfiler system that utilized a random forest classifier with time-series domain features to detect newly registered malicious domain names.

2. **Graph-based Approach**: Previous studies on graph-based approaches used the association between the domains and IP addresses or clients to form a domain graph. These studies utilized graph-based learning algorithms such as belief propagation, label propagation, and graph convolutional networks for domain classification. Khalil et al. [35] proposed a domain–IP bipartite graph, utilizing the association between the domains and IPs, followed by a path-based algorithm to discover potential malicious domains. Kazato et al. [36] introduced a graph convolutional network-based malicious domain detection method by building a domain relation graph. This approach incorporated the domain–IP relationship, domain owner information, and autonomous system number to construct the domain graph. In the graph-based domain classification approach, the association between the domains significantly influences the classification accuracy.

3.3.2 Domain Name System

Domain name system serves as one of the fundamental protocols on the Internet. DNS is a decentralized and hierarchical naming system for resources connected to the Internet. It

facilitates mapping human-readable domain names into their respective IP addresses or vice versa (IP to domain name). The DNS namespace hierarchy starts with a root domain at the top, represented by a dot(.). Under the root domain is the top-level domain (TLD), which contains the generic top-level domain (gTLD) or country code top-level domain (ccTLD). Under the TLD is the second level domain (SLD), and so on.

Fig. 3.1 illustrates the overview of how the DNS resolver handles the client request. In the figure, the client initiates a DNS request to connect to the website "duolingo.com.". The recursive resolver (DNS resolver) handles the incoming DNS request and maps the domain name to a corresponding IP address. If the recursive resolver has no cache data for the requested domain, it sends a request to the root name server, which once more routes the request to the *.com* TLD server. After that, the recursive resolver sends a direct request to the *.com* TLD server, which in turn forwards the resolver to the SLD server to retrieve the IP address of "duolingo.com.". Finally, the IP address is returned to the recursive resolver, which caches this information for future requests and returns the IP address to the client. The client can now establish a connection with the "duolingo.com." using the returned IP address.

There are four types of DNS nameservers in general: recursive resolver, root nameserver, TLD nameserver, and authoritative nameserver.

1. **Recursive Resolver**: Recursive resolver is the first nameserver to receive the client's DNS request. It resolves the domain name to the corresponding IP address using its cache data. If it has no cache data for the requested domain, the recursive resolver forwards the request to the root name server, then to a TLD name server, and eventually to the authoritative nameserver. The authoritative nameserver replies with an IP address, and the recursive resolver forwards this information to the client.

2. **Root Nameserver**: The root server accepts a recursive resolver's request, and based on the TLD extension (*.com, .org, .edu*), it forwards the recursive resolver's request to the TLD nameserver.

3. **TLD Nameserver**: This server maintains the records related to all TLD domain names. For instance, if the domain extension is *.com*, then the corresponding TLD nameserver is the *.com* nameserver. Eventually, the TLD server instructs the recursive resolver to contact the corresponding authoritative nameserver for the requested domain.

4. **Authoritative Nameserver**: A TLD nameserver forwards the recursive resolver's request to the authoritative nameserver, which maintains the IP address of the domain. If the requested domain has a CNAME (alias) record, it replies with that alias domain. At that time, the recursive resolver is required to make a new DNS request.

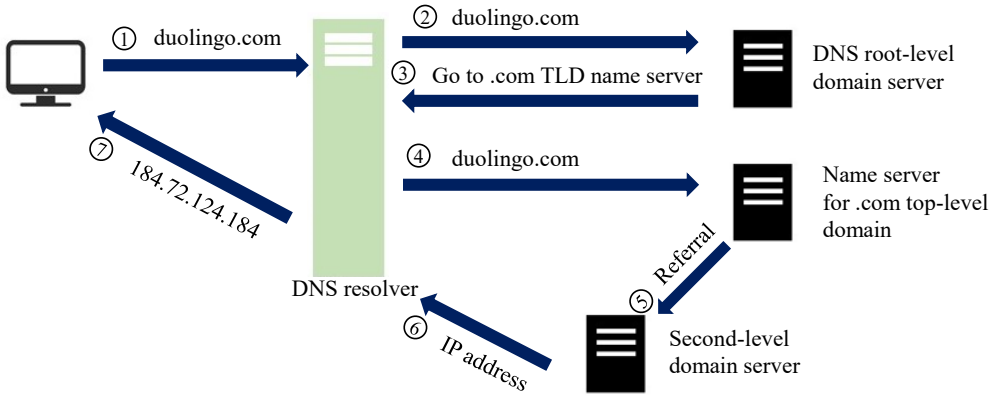

Fig. 3.1. Managing client requests in Domain Name System

3.3.3 DNS Traffic Analysis

In order to avoid domains from being blacklisted, the attackers keep moving their compromised domain names across the DNS. The two most commonly used techniques to get these behaviors are Fast-Flux and Domain-Flux (IP-Flux). In Fast-FLux, each domain name is associated with multiple IP addresses that are continuously changing to avoid blacklisting attempts. Domain-Flux utilizes the widely abused Domain Generation Algorithm (DGA) technique to dynamically create a large number of malware domain names, each associated with only one or a few IP addresses. As a result, blocking the domain names becomes challenging since most of these domains are short-lived. However, these techniques leave footprints within the DNS data. It is, therefore, crucial to analyze those traces in the DNS traffic to detect the malicious domain. The overview of how attackers utilize the domain generation algorithm to compromise the victims is illustrated in Fig 3.2.

In general, DNS traffic data can be collected in two ways: active and passive DNS data. Active DNS data are obtained by deliberately sending periodic DNS queries from the data collector. The subsequent responses were recorded for further analysis. Since the data collector initiates each query, the active DNS data does not provide any behavior of actual users, thereby easing privacy concerns [37]. The active DNS data captures the DNS records of a given domain, such as the IP address (A), name server (NS), and mail exchange (MX) records. Active DNS data do not have privacy problems because they do not include information on the user query domains. Thales [38] is an example of a privacy-preserving active DNS data collection system that actively queries and collects a large volume of active DNS data using domain names from various publicly accessible sources.

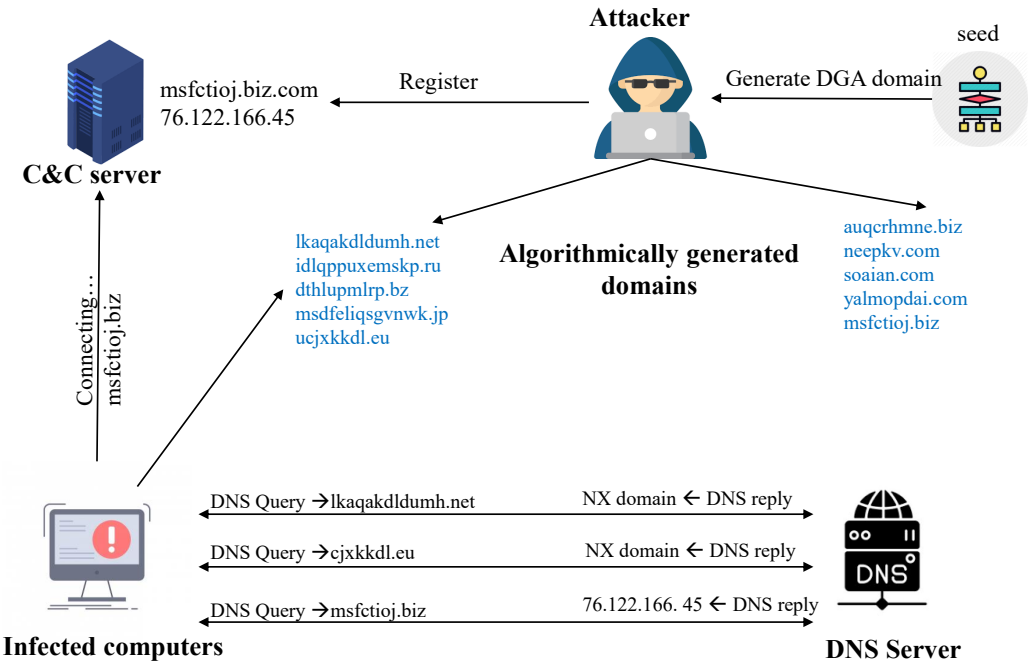

Fig. 3.2. Attackers using the Domain Generation Algorithm to compromise computer system

In contrast, passive DNS data provide historical records of the domain and contain richer information than active DNS data. Passive DNS provides the fastest means of accessing historical data that may no longer exist in the current DNS records. While the collection method of passive DNS is more complex than active DNS, several paid services offer access to passive DNS databases. Passive DNS data is gathered by deploying sensors on multiple DNS servers and DNS server logs to obtain actual DNS queries and response information. However, certain limitations and privacy issues regarding the collected data may arise depending on the placement of sensors, especially if sensors are deployed between clients and resolvers.

Kountouras et al. [38] conducted an experiment comparing active DNS with passive DNS data. They demonstrated that active DNS data has more DNS record types while passive DNS data provides a tighter connection graph for a given domain. Moreover, according to [37], active DNS data can be used to discover newly created and potentially malicious domains. Therefore, in our proposed approach for domain classification, we solely employed the DNS records of the domain from active DNS data.

3.4 Proposed Model

The overview of the proposed approach is illustrated in Fig. 3.3. The data collector module first collects the DNS data and additional information relating to the domains. After that, three

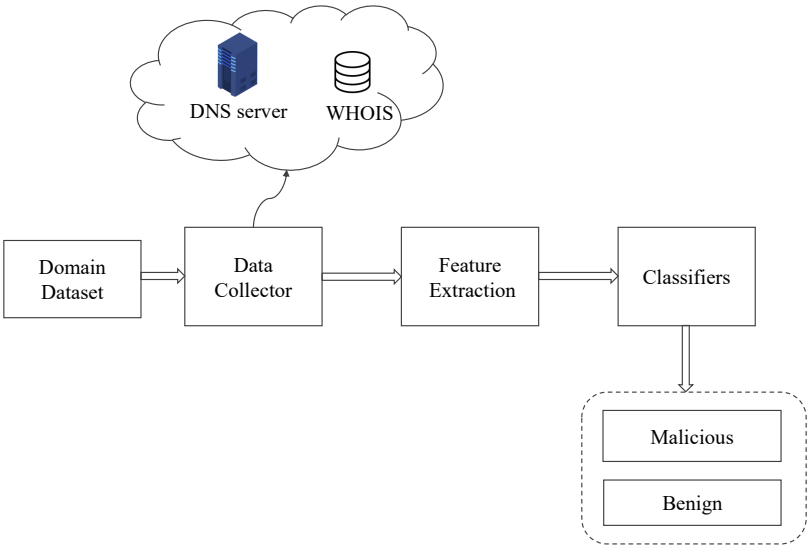

Fig. 3.3. Overview of proposed malicious domain detection model

groups of features (DNS-based, lexical, and semantic) are extracted for each domain name. The ensemble classifiers performed maliciousness estimation of the domains.

3.4.1 Data Collector

The DNS traffic data associated with each domain are queried to collect active DNS data. The DNS server then processes each query request and responds with the corresponding data. Examples of responses include the domain's A records, NS records, and TTL. This DNS response data is further enriched by the domain WHOIS information, which comprises details such as the domain registration, expiration, and updated dates. These collected DNS data are used to evaluate the maliciousness of the domains.

3.4.2 Feature Extraction

During this step, the previously collected data are processed to extract the features that can effectively distinguish malicious and benign domains. Based on the observation and analysis of the large amount of DNS data acquired from the data collector, 11 features were identified and extracted to build the classification model for malicious domain detection, as indicated in Table 3.1. The following section discusses how these features can be used to differentiate between benign and malicious domains.

(1) DNS-based Features

The DNS response records of malicious domains differ significantly from those of benign domains. Malicious domains tend to have more A (address) records and lower TTL values.

TABLE 3.1: Domain features

Type	Features
DNS-based features	Number of A records
	Number of NS records
	Number of MX records
	TTL
	Active time of domain
	Lifetime of domain
Lexical features	Number of consecutive characters
	Number of digits
	Length of domain
	Number of words
Semantic features	Domain reputation score

One of the reasons is the widespread use of the fast-flux domains [39]. The main idea behind the fast-flux is that each malicious domain is hosted on many different IP addresses, which are changed quickly to avoid being blacklisted. Moreover, a more sophisticated type of fast-flux network, known as double-flux, introduced an additional layer to make it more challenging to track malicious domains. The double-flux frequently changes both the DNS A records and NS records in a round-robin manner with a very short lifespan. As a result, DNS lookup reveals a higher count of A records and more distant NS records. Moreover, compared to benign domains, malicious domains have fewer MX records. This is mainly because domains associated with botnet attacks usually have no or fewer MX records [40], [41].

Furthermore, the lifetime and active time of benign domains are typically much longer than those of malicious domains. The lifetime of the domain is defined as $Lifetime = Date_{Expire} - Date_{Create}$, while active time is $Activetime = Date_{Update} - Date_{Create}$. The domain's lifetime is the interval between the expiration date and the domain's registration date. Similarly, the active time of the domain is the interval between the updated date and the domain's registration date. Based on these insights, the following characteristics were chosen for the DNS-based features: number of A records, number of NS records, number of MX records, TTL, active time, and lifetime of the domain.

(2) Lexical Features

Typically, benign domain name strings are readily pronounceable and easily recognizable, whereas malicious domain names tend to be non-pronounceable by humans [42], [43]. The observation and analysis of numerous malicious domains revealed that malicious domains contain a higher frequency of numbers. Moreover, the confusing mixture of numbers and words makes it difficult to pronounce malicious domains. Therefore, the following characteristics were selected for the lexical features of the domain: length of the domain, number of digits, number of words, and number of consecutive characters.

TABLE 3.2: Domain reputation scores

Domain name	Reputation score	Label
duolingo.com	44.064	Benign
discord.com	62.8	Benign
dkdrlah12.0pe.kr	7.347	Malicious
dqy.qyuyu.com	0.567	Malicious
facebook.com	63.412	Benign
douate.com	20.185	Malicious

(3) Semantic Features

The traditional approaches [6], [44] for malicious domain detection include the use of DNS data and lexical features. In this study, in addition to DNS-based and lexical features, we incorporated the semantic features of the domain. The previous study [45] introduced the detection of malicious domains using semantic features, whereby domains with the highest access rates are identified as benign domains. Each domain name is segmented by the N-gram method to create whitelist domain name substrings, which are then used to calculate a domain's reputation (maliciousness).

This study adopted a method similar to the previous approach to compute the reputation value of a domain. First, to establish the whitelist domain substring as ground truth, the top 100,000 domain names from Alexa Top Sites [46] were collected and segmented by the N-gram method. We set the lengths of N to 3, 4, 5, 6, and 7. A total of 344,503 domain name substrings were extracted from the top 100,000 domain names and used as the whitelist domain name substring. After that, the reputation score of the domain is computed by

$$\text{Reputation Score}_{\text{domain}} = \sum_{i=1}^{k} \log_2 \left(\frac{S_N(k)}{N} \right). \tag{3.1}$$

$S_N(k)$ is the total number of occurrences of the k^{th} domain name substrings in the whitelist domain name substrings. N is the length of the N-gram (N = 3, 4, 5, 6, 7). Table 3.2 presents various domain reputation scores. It is evident that the reputation score of the benign domain tended to be higher than that of the malicious domain, mainly due to the frequent occurrence of segmented benign domain substrings in the whitelist domain name substrings.

3.5 Evaluation

We evaluated the performance of the proposed model on three ensemble classifiers: random forest, XGBoost, and AdaBoost. Initially, we collected and labeled the publicly available benign and malicious domain names to make the dataset. After that, the dataset is divided into training and testing data to evaluate the effectiveness of the proposed model.

TABLE 3.3: Experimental results for malicious domain detection

Features	Classifiers	Accuracy	Precision	Recall
DNS	Random forest	0.8973	0.8955	0.8824
	AdaBoost	0.9007	0.8741	**0.9191**
	XGBoost	0.9007	0.8794	0.9118
DNS+Lexical	Random forest	0.9041	0.9033	0.9050
	AdaBoost	0.9096	0.9092	0.9113
	XGBoost	0.9068	0.9063	0.9084
DNS+Lexical+ Semantics	Random forest	**0.9270**	**0.9199**	**0.9219**
	AdaBoost	**0.9151**	**0.9146**	0.9168
	XGBoost	**0.9123**	**0.9115**	**0.9131**

Dataset: The dataset contained a total of 1,457 domain names, comprising 680 malicious domains and 777 benign domains. The benign domains were gathered from Alexa Top Sites [46], which ranks websites based on their popularity. We assumed that the top-ranked websites were legitimate domains. The malicious domain names were collected from publicly published domain blacklist services. The DNS-resolvable malicious domains were randomly selected from malwaredomainlist.com [47] and a compromised domain list [48]. These domains were known to be compromised by malware, command and control communication, and phishing activities.

Evaluation metrics: We computed the number of true positive (TP), true negative (TN), false positive (FP), and false negative(FN). FP indicates the number of normal data incorrectly classified as an attack, while FN represents the number of attack samples incorrectly classified as normal. TP and TN denote the number of normal or attack data that were accurately classified. Based on these values, the evaluation metrics, such as precision and recall are computed as follows.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \qquad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}. \tag{3.2}$$

3.5.1 Results

In the experiments, three ensemble models – random forest, AdaBoost, and XGBoost – are employed to measure the effectiveness of the proposed features in the identification of malicious domains. We conducted three experiments. Each experiment is based on the different combinations of feature sets: (i) using only DNS features, (ii) combining DNS and lexical features, and (iii) integrating DNS, lexical, and semantics features. The experimental results are indicated in Table 3.3.

In our experiment, the domain dataset was divided into 75% training data and 25% testing data. All classifiers were trained and evaluated using 10-fold cross-validation. Despite a relatively small domain dataset, all three classifiers consistently achieved above 89% accuracy in detecting

malicious domains across all experiment scenarios. The experimental results highlighted that utilizing the combination of all feature sets outperformed the scenarios that utilized only DNS features or the combination of DNS and Lexical features. Notably, random forest exhibited the best performance across all evaluation metrics when all feature sets were incorporated into the experiment. Moreover, the empirical results revealed that integrating DNS and lexical features indicates a slightly higher model performance than just using DNS features.

3.6 Conclusion

We have proposed an approach to classify a domain as malicious or benign by leveraging active DNS traffic data and WHOIS information. Moreover, we incorporated semantic features, in addition to the commonly used lexical and DNS-based features, with the aim to improve the detection of malicious domains. The experimental results demonstrated that the proposed approach achieved an accuracy of up to 93% with the random forest classifier. However, the current model is limited to identifying domains as either malicious or benign. To further provide details on the malicious domains, we can categorize each malicious domain as spam, phishing, command and control, or malware, thereby making it into a multiclass classification problem. Moreover, using a combination of passive DNS and active DNS data could enhance the ability to detect bad domains.

Chapter 4

Few-Shot Learning-Based Malicious IoT Traffic Detection with Prototypical Graph Neural Networks

4.1 Abstract

With a rapidly escalating number of sophisticated cyberattacks, protecting Internet of Things (IoT) networks against unauthorized activity is a major concern. Detecting malicious network traffic is thus crucial for IoT security to prevent unwanted traffic. However, existing malicious traffic detection systems that rely on a supervised machine learning approach need a considerable number of benign and malware traffic samples to train the machine learning models. Moreover, in the cases of zero-day attacks, only a few labeled traffic samples are accessible for analysis. To deal with this, we proposed a few-shot malicious IoT traffic detection system with a prototypical graph neural network. The proposed approach does not require prior knowledge of network payload binaries or network traffic signatures. The model is trained on labeled traffic data and tested to evaluate its ability to detect new types of attacks when only a few labeled traffic samples are available. The proposed detection system first categorizes the network traffic as a bidirectional flow and visualizes the binary traffic flow as a color image. A neural network is then applied to the visualized traffic to extract important features. After that, using the proposed few-shot graph neural network approach, the model is trained on different few-shot tasks to generalize it to new unseen attacks. We evaluated the proposed model on a network traffic dataset consisting of benign traffic and traffic corresponding to six types of attacks. The results revealed that our model achieved an F1 score of 0.91 and 0.94 in 5-shot and 10-shot classification, respectively, outperforming the baseline models.

4.2 Introduction

The Internet of Things (IoT) has become one of the fastest-growing technologies in recent years. With an increase in the extensive use of IoT devices in fields such as healthcare, smart homes, smart cities, manufacturing, and the automotive industry, the importance of Internet-connected devices in daily life has increased. In addition, with the increasing number of cyber-attacks, it is critical to ensure that IoT devices are protected from potential cyber threats. However, most IoT devices are known for their vulnerabilities and lack of on-device security controls to defend

against cyber-attacks [49–51]. This is because most IoT devices have resource constraints and insufficient computing power, which allows only limited functions to be executed [51]. Attackers exploit loopholes in IoT devices to perform malicious activities, such as using IoT devices for botnet attacks. One example is the Mirai [52] botnet attack that occurred in 2016 and caused massive Internet security breaches by exploiting insecure IoT devices.

To protect IoT networks from cyber-attacks, intrusion detection systems [9, 10, 53] have been used extensively to monitor unauthorized activities and identify notorious network attack traffic in the IoT ecosystem. Depending on the technique employed, intrusion detection systems can be mainly categorized into signature-based and anomaly-based systems. The former systems compare incoming traffic to predefined attack signatures in a database through pattern matching. This technique is effective and highly accurate for previously seen (known) attacks. However, it is ineffective in zero-day attack detection since the new attack signatures are not in the database yet. Artificial intelligence enables anomaly-based intrusion detection systems to overcome the limitations of the signature-based approach. The anomaly-based approach recognizes any behavior that deviates from the observed norm as an anomaly. With recent advancements in machine learning (ML), many studies have applied ML algorithms for the detection of malicious attacks in the IoT network monitoring system (e.g., [9, 10, 53]). Providing that there is a considerable number of labeled samples for training the ML models, the detection systems can distinguish normal behavior from abnormal behavior with relatively high accuracy.

However, there are a couple of issues in ML-based IoT intrusion detection systems that must be addressed. The amount of available benign traffic is enormous compared to abnormal malicious traffic; moreover, some IoT malware attack types have significantly fewer samples. As a result, the collected IoT dataset can be highly imbalanced [54]. Most importantly, new IoT malware threats are constantly emerging; subsequently, ML-based detection systems face a new challenge known as zero-day attacks [55], which are types of attacks that do not exist at the time of model training. It may take time for security vendors and researchers to publish a huge number of samples of the new threats; therefore, there is a need for a system that can detect IoT network traffic by using a limited number of new malicious samples. This can be effectively accomplished through a few-shot learning method. Few-shot learning is a type of ML method that aims to make predictions with only a few labeled data in a supervised setting. The previous studies [56–59] on few-shot learning used the principle of meta-learning, which enables a certain number of correlated tasks to be learned during meta-training. In the meta-testing stage, the trained learner can be utilized to predict unseen but related tasks by providing only a few labeled samples.

To defend against potential cyber-attacks in the IoT ecosystem, malicious IoT traffic detection systems are crucial. In this study, we propose a detection model based on the binary visualization of network traffic and few-shot learning to detect unknown malware attacks on IoT networks. The rationale for representing network traffic as an image is that a malware traffic image

exhibits significantly more clustered patterns than a benign traffic image, which exhibits a more consistent and static pattern [60]. Furthermore, representing the captured IoT traffic as an image provides a clearer comprehension of the network traffic since the same type of malware attack generates similar image patterns [61] that can be recognized by a deep learning model. This approach eliminates the need for manual feature engineering and prior domain knowledge. According to [62] and [63], the deep convolutional neural network model performs better on image datasets. The authors transformed the network traffic dataset into a three-dimensional image and demonstrated that the neural network model trained upon the image-based network traffic dataset is superior to conventional machine learning methods. Following these approaches, in our study, we first transform the collected network traffic into the image dataset. After that, we develop a model that can identify malicious IoT network traffic based on the network image dataset and a few-shot learning-based prototypical graph neural network. Our main contributions are as follows.

1. We propose a few-shot learning-based IoT network traffic detection system using binary visualization and the prototypical graph neural network. Our model can detect and identify new malware attack traffic using only a few labeled samples at the meta-testing stage.

2. We convert the network traffic into a red–green–blue (RGB) image to build the dataset for the malware traffic image. After that, a pre-trained convolutional neural network (CNN) on the ImageNet dataset is employed to extract the features.

3. We conduct an extensive experiment to demonstrate the effectiveness of our proposed model. The experimental results indicate that our model can detect new IoT network attacks with a few labeled samples and is applicable to the detection of a diverse range of malware attacks.

4.3 Background

4.3.1 Network Intrusion Detection System

The network Intrusion Detection System (IDS) plays a crucial role in the field of cybersecurity. It monitors, detects, and responds to unauthorized activities in the computer network. The primary role of IDS is to keep track of the network activities, analyze those activities, check the potential vulnerabilities, identify attack signatures or patterns, detect network anomalies, and generate alerts if the anomalies are discovered. The main components of IDS are illustrated in Fig 4.1, and detailed explanations are provided as follows.

1. **Network Monitoring and Data Collection**: It is necessary to continuously monitor the computer network to collect information related to network activities since the attackers can perform the attacks by injecting malicious code or exploiting the vulnerabilities.

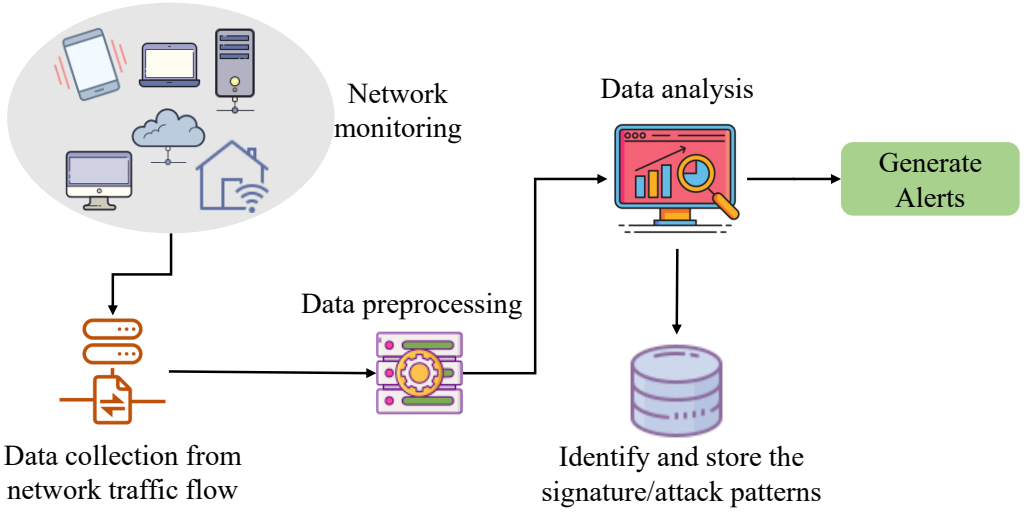

Fig. 4.1. Network intrusion detection system

Generally, the multiple network packets, which are the combination of packet header and payload, can be gathered in this component. Both header and payload offer valuable insights into the potential attack activities. Moreover, we can also collect information related to the network flow, which is a sequence of packets grouped by five attributes: source IP address, destination IP address, source port, destination port, and protocol. We can build the network dataset and extract the packet-level and flow-level features from the collected data to identify the network attacks.

2. **Preprocessing**: To apply the machine learning/deep learning techniques, the raw network data should be preprocessed to extract important features. Techniques such as data normalization, scaling, and handling of missing or null values are performed in this step.

3. **Analysis**: This step involves the detailed analysis of network packets to identify the known attack signatures or patterns. These signatures are stored in the database to recognize future attack behavior through pattern matching. Recently, with the emergence of artificial intelligence, machine learning, and deep learning approaches have become mainstream techniques for classifying anomalous network behavior. Given that the signature-based approach is limited to identifying the known attack patterns, the incorporation of machine learning techniques offers an effective way to recognize unknown attack signatures.

4. **Generating Alert**: After identifying the attack signatures/patterns or detecting the network anomaly, an intrusion alert is generated and sent to the CERT/CSIRT team. The security experts are responsible for making decisions on how to mitigate the cyberattacks and prevent future attacks.

4.3.2 Signature-based vs. Anomaly-based IDS

Signature-based and anomaly-based approaches are the two main techniques utilized to identify threats in the network intrusion detection system. In the signature-based IDS approach, a series of known attack signatures or patterns are already predefined in the database. This approach monitors the packet traversing the network and looks for intrusion events that match the already existing signatures. It is an effective and highly accurate way to recognize the previously known attacks. However, it is ineffective in zero-day attack detection since the new attack signatures are not in the database yet, and thus, it fails to find the correlation between the new attack patterns and existing known attack patterns.

On the other hand, the anomaly-based IDS approach utilizes machine learning techniques to train the anomaly detection system. Rather than searching for the known signatures, the anomaly-based approach compares the network activity with normal network patterns and generates intrusion alerts if abnormal behavior is detected. The advantage of anomaly-based IDS is zero-day attacks can be recognized by analyzing patterns, as slight deviations from normal behaviors are considered anomalous. One disadvantage of this approach is that normal behaviors can be wrongly recognized as abnormal, leading to multiple false positives (false alarms) and subsequently increasing the workload on the security experts to investigate all these alerts. Therefore, it is better to use the combination of both approaches to leverage the strengths of each approach and enhance the overall detection capabilities.

4.3.3 Few-shot Learning

With the advancement of technologies in recent years, artificial intelligence has been in the limelight due to its efficiency and human brain-like capability. One drawback of traditional machine learning is that it requires enormous data to train the desired tasks. To learn the model from the limited training data, a new machine learning paradigm called federated learning is introduced [64]. In general, few-shot learning aims to recognize new unobserved tasks using a small quantity of labeled training samples.

The N-way K-shot is the expression often used in few-shot learning. K-shot indicates the number of samples per class used in the model learning. If K is equal to 1, this scenario is known as one-shot learning, where the model is trained with just one sample per class for each training task. Various few-shot learning approaches utilize the meta-learning framework with episodic training [56–59]. In these approaches, diverse few-shot tasks are sampled from the meta-train dataset for each training episode during the model training. Each few-shot task is made up of a known support set and an unknown query set. It is important to note that the classes in the query set might never overlap with the support set. Instead of determining which class each sample belongs to, the few-shot meta-learner learns the similarity or dissimilarity between the support and query sets to assign label information from a support instance to a

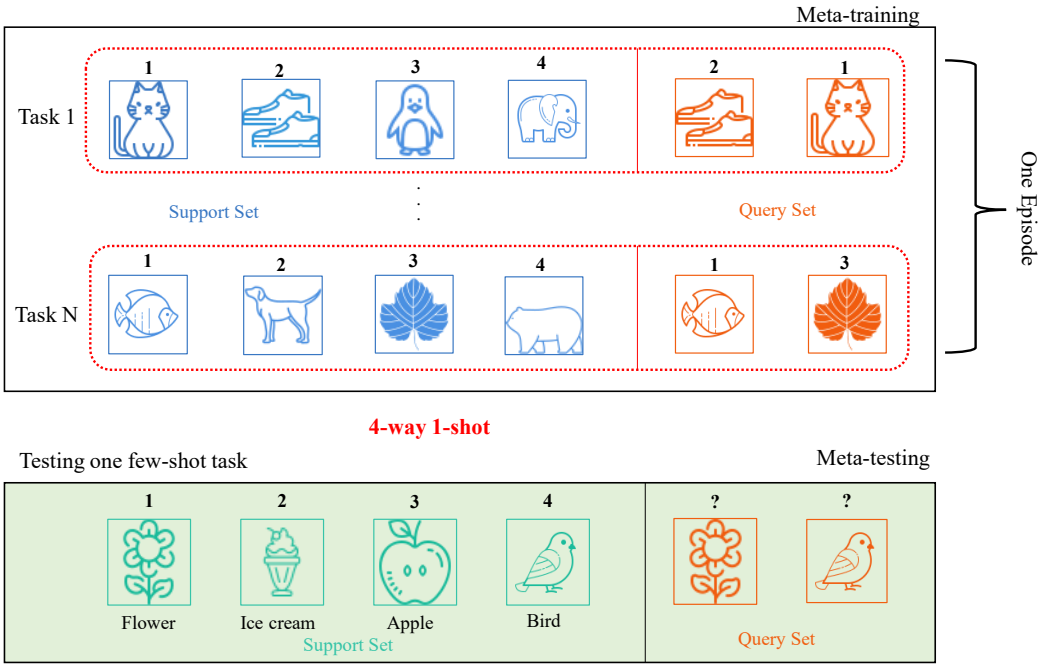

Fig. 4.2. Few-shot Learning

query instance. Fig. 4.2 illustrates the tasks involved in the meta-training and meta-testing stages of few-shot learning.

The authors of [56–58] presented few-shot learning paradigms that compared labeled support set and unknown query samples in a shared embedding space to predict the label information for query instances. Matching Networks [56] utilizes the weighted nearest neighbor search with the attention mechanism, while Relation Networks [57] trains the model to learn the nonlinear relation between the support and query sets. Prototypical Networks [58] calculates the prototype representation of each class in the support set in the embedding space, and the label of the query sample is predicted by computing the Euclidean distance. The graph-based few-shot network traffic detection method proposed in this paper follows the principle of Prototypical Networks.

4.3.4 Graph Neural Networks for Few-shot Learning

Artificial neural networks are capable of capturing hidden patterns in Euclidean data, such as images, text, and audio. However, some data may have underlying graph structures with complex relations and interdependencies. For example, the graph structure can be found on social networks, knowledge graphs, and protein interaction networks. This has led to the advancement of the Graph Neural Networks (GNNs). GNNs are a deep learning architecture designed to analyze graph-structured data [65]. A graph consists of a set of vertices (nodes) joined by edges. The graph neural network learns the current node representation by repeatedly combining the features of neighboring nodes using the message-passing algorithm, thereby

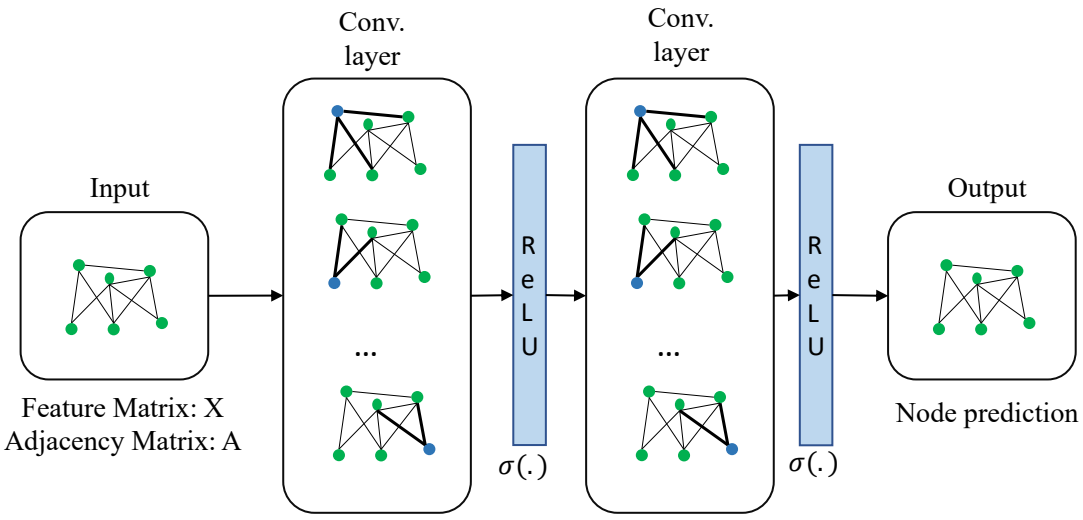

Fig. 4.3. Graph Convolutional Neural Networks

generating similar representations for strongly linked nodes. A graph convolutional neural network, a well-known variant of GNNs, introduced by Kipf et al. [66] for semi-supervised classification is depicted in Fig 4.3.

Due to the advantages of GNNs, some approaches [59, 67, 68] incorporated GNNs in the few-shot learning domain and demonstrated promising results. In their GNNs, each support or query sample is represented by a graph node. Each graph node has an attribute created through the concatenation of the sample's (support or query) feature embedding and label embedding. The final classification layer is designed to predict the class probability of each query node.

The authors of the previous study [59] utilized a GNN as a label propagation module to forecast the label of unlabeled nodes. In addition, the edge-labeling GNN (EGNN) for few-shot learning proposed in [67] predicted the edge labels between the support and query sets by iteratively updating the node and edge features to ensure intra-cluster similarity and inter-cluster dissimilarity. Furthermore, the fuzzy GNN (FGNN)[68] employed the fuzzy membership function to update the edge labels iteratively. After that, node classification was performed on the constructed graph to predict the unlabeled nodes.

4.3.5 Existing Study on IoT Network Traffic Detection

With the growing number and accelerated use of IoT devices, their vulnerabilities have become a target for cybercriminals, contributing to a surge in cyberattacks and information leakage. As discussed below, security experts in the research community have dedicated vigorous efforts to address security and privacy concerns in IoT networks.

In previous studies [9, 10], ML was the most commonly used approach to distinguish malicious traffic intrusion in IoT networks. The intelligent integrated intrusion detection system proposed by [9] used a deep learning algorithm to discover malicious attacks in real IoT network traffic. After separating the incoming traffic into sessions, features such as the source and destination IP address, transmission mode, duration, transmission and reception rate, and transmission-to-reception ratio were extracted and forwarded to a deep neural network. An average precision of 95% and recall of 97% were achieved for five attack scenarios: blackhole, sinkhole, wormhole, distributed denial-of-service (DDoS), and opportunistic service attacks.

In [10], the authors addressed cyber threats in a smart city infrastructure by proposing the random forest classifier-based anomaly detection system in fog nodes. The authors claimed that the proposed model could effectively detect compromised IoT devices. The classification results on the UNSW-NB15 dataset indicated that the model predicted the normal class with an F1 score of 0.99 and attack traffic with an F1 score of 0.86.

Some studies [69], [63] converted network traffic into an image based on the network packets, flow, and session and applied a convolutional neural network (CNN) model to the produced images to identify malicious traffic. In [69], the authors proposed a malicious IoT traffic classification technique. They transformed network traffic as an image and utilized ResNet50 to analyze the visualized data. The model was evaluated on a dataset of 1,000 PCAP files of benign and malicious traffic and demonstrated promising results. Similar to [69], the methodology proposed in [63] transformed traffic data packets and flow information into images. Then, the resulting image dataset is classified with the residual neural network (ResNet) model. The empirical results indicated the multi-class classification with the CICDDoS2019 dataset achieved an F1 score of 0.86.

Network intrusion detection systems based on the few-shot learning paradigm have also been proposed in recent studies [70, 71]. These previous studies have effectively demonstrated the capability of the trained meta-learner to detect new attacks using just a few labeled data. Section 4.4.1 provides a detailed explanation of the meta-learning-based few-shot framework.

The system proposed in [70] consists of two main parts: deep-neural-network-based feature extraction and feature comparison. The feature extraction part produces feature map pairs for network traffic sample pairs, while the comparison part computes the delta score. The comparison result indicates whether a pair of traffic samples belong to the same class. A few-shot dataset was constructed from the ISCX2012 and CICIDS2017 datasets, and the authors showcased that malicious traffic could be detected up to 99%.

In [71], the authors proposed an approach that performs feature embedding of traffic samples with a trained embedding function and calculates the cosine distance between the samples to predict which traffic samples are closer in the embedding space. The experimental results

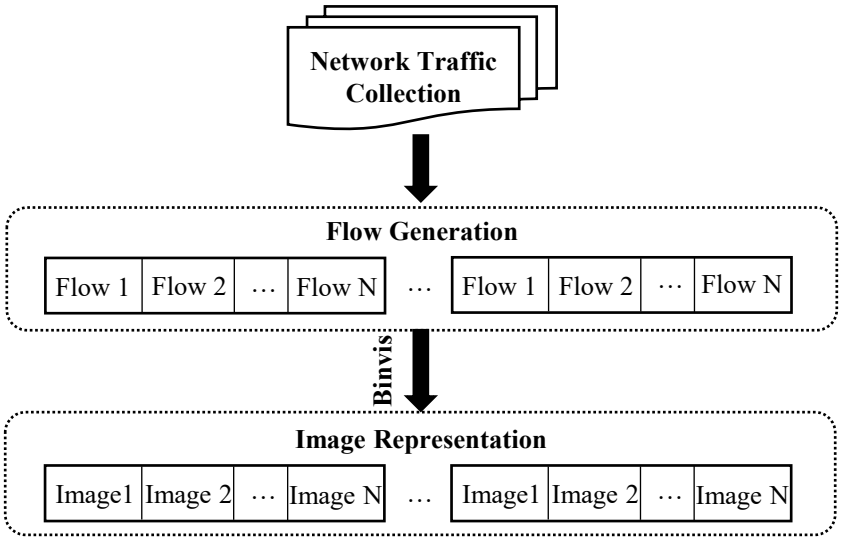

Fig. 4.4. Network traffic preprocessing

indicate that the choice of the embedding and distance functions affects the accuracy of the classification model.

Our study differentiates itself from existing studies by specifically examining capabilities and challenges encountered in applying graph-based few-shot learning to the intrusion detection domain. Moreover, the transformation of network flow information to the RGB image dataset by utilizing Binvis [72] is also a novel way in the domain of the few-shot learning-based intrusion detection system.

4.4 Proposed Method

4.4.1 Few-shot Learning Strategy

The proposed few-shot learning approach follows the Prototypical Networks described in [58], which utilizes the meta-learning framework with episodic training. Few-shot learning is also known as M-way K-shot classification. M-way denotes how many classes are in each task T, while K-shot signifies that each class has K samples. Like traditional supervised learning with a training and test dataset, few-shot learning has a meta-training set M_{Train}, and a meta-test set M_{Test}.

Rather than training on the entire training dataset at once, the meta-learner is trained to learn over diverse few-shot tasks T in multiple episodes. For every task T, T is made up of support set S and query set Q, and the samples in both sets are of the same class. The support set is labeled, while the query set is unlabeled and needs to be predicted. Therefore, for every episode at the meta-training stage, we sample the M-way K-shot task T from dataset M_{Train} as $S_T = \left\{(x_j, y_j) \,|y_j \varepsilon C, j = 1, ..., K \times N_S\right\}$, $Q_T = \left\{(x_k, y_k) \,|y_k \varepsilon C, k = 1, ..., K \times N_Q\right\}$, and

TABLE 4.1: Color mapping by the Binvis binary data visualization tool

Color	Description of ASCII characters
White	0xFF
Black	0x00
Blue	Printable
Green	Control
Red	Extended

$S_T \cap Q_T = \varnothing$. The symbol C denotes the set of train classes and N_S, N_Q represent the number of samples for each class in the support set and query set, respectively.

During the training phase, the meta-training task $T_{\text{Train}} = (S_T, Q_T)_{T=1}^{i}$ is trained by i number of episodes with known label information for both the support and query sets. The trained model is then used to predict the label of the query sample of M_{Test}, with a few labeled support samples. The M-way K-shot task T for the meta-test dataset is prepared in the same way as the meta-training task. Ideally, the classes used in the meta-training and meta-testing phases are entirely different, which achieves the goal of predicting zero-day IoT network traffic attacks with limited labeled data.

4.4.2 Data Preprocessing

Because the proposed model represents IoT network traffic as an image, some data preprocessing must be carried out on the raw IoT network traffic. The data preprocessing step is illustrated in Fig. 4.4. First, the individual bidirectional flows are separated from the collected network traffic file. A network flow is a group of several associated packets [73] grouped by the 5-tuple: source IP address, destination IP address, source port, destination port, and protocol. Each flow contains the hexadecimal sequence of the time-adjacent packets identified by the same 5-tuple.

After that, the hexadecimal values in each flow are transformed into the RGB color image using a binary data visualization tool called Binvis [72]. Binvis maps each hexadecimal value in a flow to a predefined color conversion scheme, as illustrated in Table 4.1. The generated output is the one-dimensional color sequence of each flow. The next step is to lay out each generated color sequence as an RGB color image while preserving the proximity of the elements in the one-dimensional sequence to be as near as possible in the two-dimensional image. This can be achieved by means of the Hilbert space-filling curve [74]. After positioning the color sequence of the network flow in a two-dimensional layout, the desired RGB color image can be produced. When visualizing the individual flow as an image, we consider the whole packet (i.e., both header and payload). In our approach, each IoT network traffic flow is visualized as a color image of size 256×256. Examples of visualized network traffic are presented in Fig. 4.5. One advantage of visual representation is that it provides a clearer view and comprehensive understanding of the overall network traffic since the same malware traffic families tend to generate similar image patterns.

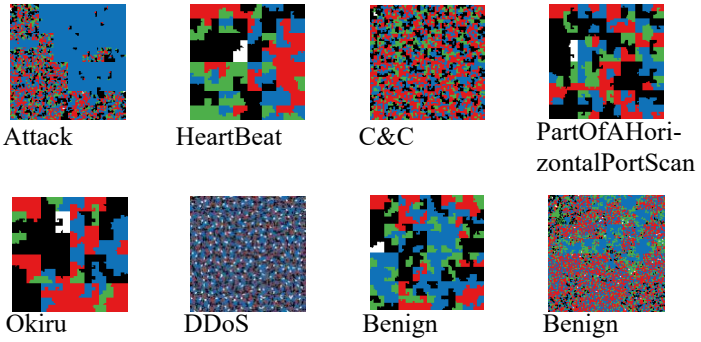

Fig. 4.5. Visual representation of network flows

4.4.3 Few-shot Prototypical Graph Neural Network

An overview of the proposed few-shot GNN model is illustrated in Fig. 4.6. The architecture consists of four main parts: feature extraction, a prototype encoder, a graph construction module, and a graph neural network classifier.

Feature Extraction: The feature embedding function aims to extract important features in the embedding space. Various CNNs or deep neural networks can be used to extract features. The proposed model utilizes a pre-trained ResNet18 [75] as the feature embedding function. ResNet18 is a CNN model that consists of 18 convolution layers. After removing the final fully connected layer intended for classification, the remaining layers in ResNet18 can be regarded as the feature embedding module. The feature extractor applied in our proposed scheme is pre-trained on the ImageNet dataset [76]. The dimension of the output features vector is 512.

Prototype Computation: After feature embedding, the prototype of each class is computed with the label information from the support set. The approach described in [58] is used to obtain the prototype of each class by obtaining the mean of the support set's feature embeddings associated with that class. The following equation calculates the prototype belonging to class c:

$$Proto_c = \frac{1}{|S_c|} \sum_{(x_j, y_j) \varepsilon S_c} f_\theta \left(x_j \right), \tag{4.1}$$

where $f_\theta \left(x_j \right)$ is the feature embedding of the j^{th} class from the support set, while S_c represents the set of support samples belonging to class c. Next, the Euclidean distance is calculated, which is the distance between the computed class prototype and the query instance. The minimum distance is chosen as the initially predicted label for the query instance. **Graph Construction**: Following the computation of the initial labels for the query set instances, a graph consisting of support set images and query set images is constructed. Generally, a graph is formed by a group of nodes with a link (edge) between nodes. In the proposed scheme, we consider images from both the query and support sets as nodes. Since the labels of the instances in the support set are already known, an edge is added between support nodes if they are from the same class.

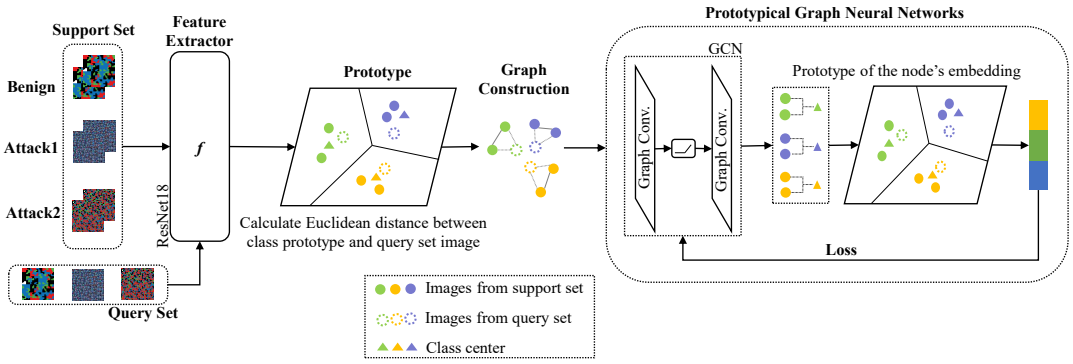

Fig. 4.6. Proposed few-shot learning-based network traffic detection system

The relations between the support and query nodes are obtained from the previously computed Euclidean distance. If the initially predicted class of the query node and the class of the support nodes are the same, there is a relation (edge) between the nodes. The constructed graph and the concatenated feature embeddings of both sets are forwarded to the GNN classifier.

Prototypical Graph Neural Networks: The proposed method employs a graph convolutional network (GCN), a variant of a GNN proposed in [66], as the graph classifier. Similar to a CNN, which is commonly used in computer vision, a GCN performs multilayer convolution on graph-structured data. A GCN can be used for link prediction, graph classification, and node classification. In the proposed approach, node classification is applied to the graph constructed in the previous step to predict the label information of the unlabeled query nodes. The output of the graph classifier is the predicted labels of the query set.

The multilayer GCN utilized in our proposed method for node classification is explained in the following section. For a given graph G with vertices V and edges E, the input of the GCN is the node features, adjacency matrix, and label information. The output of the GCN is the node classification results, which predict the unlabeled nodes. A multilayer GCN [66] defines the layer-wise propagation rule as follows:

$$H^{(l+1)} = \sigma\left(\tilde{D}^{(-\frac{1}{2})}\tilde{A}\tilde{D}^{(-\frac{1}{2})}H^{(l)}W^{(l)}\right). \tag{4.2}$$

For an undirected graph $G = (V, E)$ with nodes $v_i \varepsilon V$ and edges $(v_i, v_j)\ \varepsilon E$, let $A \varepsilon R^{N\times N}$ and $X \varepsilon R^{N\times C}$ be the adjacency matrix and feature vector of G, respectively, and let $D = \sum_j A_{ij} \varepsilon R^{N\times N}$ be the degree matrix of A, where $i, j = (1, ..., N)$, N is the total number of vertices V, and C is the dimension of the feature vector. In Eq. 4.2, $H^{(l)} \varepsilon R^{N\times D}$ is the feature vector that is input to the l^{th} layer of the GCN. Therefore, we can say that H^0 is identical to the input feature vector X. Here σ denotes the activation function, and $W^{(l)}$ is the layer-specific trainable weight matrix of the l^{th} hidden layer. The notation $\tilde{D}^{(-\frac{1}{2})}\tilde{A}\tilde{D}^{(-\frac{1}{2})}$ is the normalized adjacency matrix with

a self-loop, where $\tilde{A}$ is the identity matrix of A, and $\tilde{D}$ is the degree matrix of $\tilde{A}$. The GCN performs aggregation, combining steps recursively in each layer. Each node calculates the mean value of the neighboring nodes' features and its own features. The aggregated feature values are then multiplied by weight W, and the status of every node in each layer is updated using the ReLU activation function. W is updated by the minimum cross-entropy loss function.

The prototypical graph neural network model used in the proposed approach is similar to [66]. Our graph model is made up of two parts: two-layer graph convolution and prototype computation of support node embedding. After the graph convolution layer computed the node embeddings for each support and query node, we calculate the prototype of the support node embeddings. The mean value of the support node embeddings belonging to the same classes is calculated as the node prototypes (i.e., if the number of the support node class is four, then the total node prototype is four). Then, we find the Euclidean distance between the node prototypes and the query node embeddings. The graph classifier finally outputs the predicted class of the query nodes based on the nearest distance between the node prototypes and the query nodes.

4.4.4 Training Objectives and Parameters

During the meta-training, the model is optimized with two cross-entropy loss functions to make predictions for each query sample using its respective class prototypes. The first loss function for the prototype computation module is computed via cross-entropy loss: $L_1(\alpha) = -\sum \log P_\alpha(y|x, S_T)$, which is the negative log-likelihood of the true class of each query sample. Similarly, the Prototypical Graph Neural Networks module is optimized again with the cross-entropy loss function: $L_2(\gamma) = -\sum \log P_\gamma(y|x, S_T)$. P_α and P_γ denote the class probabilities of the query sample in each episode over parameters α and γ. It can be calculated by taking the softmax over the Euclidean distance between the query sample x and each class prototype P_c as:

$$P_\alpha(y = c|x) = \frac{\exp(-\mathrm{ED}(f_\alpha(x), Proto_c))}{\sum_{c'} \exp(-\mathrm{ED}(f_\alpha(x), Proto_{c'}))}, \tag{4.3}$$

where, ED represents the euclidean distance and $P_{c'}$ is the prototype of the class c'. The calculation of P_γ is similar to P_α. The total loss during meta-training is optimized by $Loss = L_1(\alpha) + L_2(\gamma)$, and the model parameters are updated by minimizing the total loss. The complete training algorithm for one few-shot task is provided in Algorithm 1.

The proposed prototypical graph neural network model consists of 2 hidden graph convolution layers with 8 hidden units. A dropout layer of 0.002 is applied in the graph model to avoid over-fitting. The proposed model utilizes the Adam optimizer with a learning rate of 0.001 and 5e-5 weight decay. The model parameters are determined by tuning the hyperparameters.

Algorithm 1 The training algorithm for one few-shot task

Input: Meta-train dataset $M_{\text{train}} = \left\{\left(x_j, y_j\right) | y_j \varepsilon C\right\}$, C: the set of training classes, N_C: the number of classes in one few-shot task, N_S: the number of support samples per class, N_Q: the number of query samples per class

Requires: Feature extractor f

- ➢ $RANDOMSAMPLE(M_{\text{train}}, N)$ represents a set of N examples chosen randomly from M_{train} without replacement.
- ➢ *GCN(g)* denotes the graph convolutional neural network with input graph g.
- ➢ $GRAPH\left(S_T, Q_T, \text{img\_dist}\right)$ constructs each image in S_T and Q_T as a node. The img_dist is the edge between the support node and the query node. The edges between the support nodes are obtained from the label of the support set.

Output: Loss L for backpropagation

$L \leftarrow 0$ ▹ initialize loss
for `T in` $\{1, 2, ..., N_C\}$ **do**
  $S_T \leftarrow RANDOMSAMPLE(M_{\text{train}}, N_C \times N_S)$ ▹ select the support samples set
  $Q_T \leftarrow RANDOMSAMPLE(M_{\text{train}}\ S_T, N_C \times N_Q)$ ▹ select the query samples set
end for

`(1) Compute support feature` $f_\theta(x_j)$ `and query feature` $f_\alpha(x_j)$ `from` S_T `and` Q_T
for `c in` $\{1, 2, ..., N_C\}$ **do**
  $Proto_c \leftarrow \frac{1}{|S_c|} \sum_{(x_j, y_j)\varepsilon S_c} f_\theta\left(x_j\right)$ ▹ compute image prototype for class c
end for

$\text{img\_dist}_c = \min\left(\text{euclidean\_dist}\left(f_\alpha\left(x_j\right), Proto_c\right)\right)$ ▹ calculate Euclidean distance
$L_1(\alpha) = -\sum \log P_\alpha\left(y|x, S_T\right)$, where, ▹ loss for image prototype computation

$$P_\alpha\left(y = c|x\right) = \frac{\exp\left(\text{img\_dist}_c\right)}{\sum_{c'} \exp\left(\text{img\_dist}_{c'}\right)}$$

$g = GRAPH(S_T, Q_T, \text{img\_dist}_c)$ ▹ construct graph
$out = GCN\left(g\right)$ ▹ (2) compute two-layer graph convolution on graph g

`Compute the support node embedding` $f_{g\theta}\left(x_j\right)$ `and query node embedding` $f_{g\alpha}\left(x_j\right)$ `from` *out*
for `c in` $\{1, 2, ..., N_C\}$ **do**
  $nProto_c \leftarrow \frac{1}{|S_c|} \sum_{(x_j, y_j)\varepsilon S_c} f_{g\theta}\left(x_j\right)$ ▹ compute node prototype for class c
end for

$\text{node\_dist}_c = \min\left(\text{euclidean\_dist}\left(f_{g\alpha}\left(x_j\right), nProto_c\right)\right)$
$L_2(\gamma) = -\sum \log P_\gamma\left(y|x, S_T\right)$, where, ▹ loss for graph prototype computation

$$P_\gamma\left(y = c|x\right) = \frac{\exp\left(\text{node\_dist}_c\right)}{\sum_{c'} \exp\left(\text{node\_dist}_{c'}\right)}$$

$L \leftarrow L + L_1 + L_2$ ▹ optimize loss

TABLE 4.2: Statistics of image dataset used in experiment

Class label	Number of flows
Benign	4,871
Attack	1,327
Heartbeat	3,678
C&C	2,139
DDoS	28,041
Okiru	22,409
PartOfAHorizontalPortScan	21,033
Total	83,498

4.5 Evaluation

This section evaluates and discusses the efficiency of our proposed few-shot learning-based IoT network traffic detection method. The IoT-23 dataset [77] is used in the experiment to demonstrate the efficiency and performance of the proposed model. Moreover, we compared the proposed method with two baseline methods: Prototype Networks [58] and FGNN [68]. The proposed model is trained on NVIDIA Quadro RTX 5000 with 16GB memory.

Dataset: The experiment in this study was conducted on the IoT-23 dataset, which consists of 20 malware traffic captures and three benign traffic captures, collected from 2018 to 2019. The benign traffic was gathered from a Philips Hue LED lamp, Amazon Echo, and Somfy smart door lock, while the malicious traffic was generated by simulating various botnet attacks on those IoT devices. The dataset includes the original captured network files (in PCAP format) and the log files, generated by the Zeek network analyzer [78], with a total of 21 feature attributes, including the label information. The dataset is made up of approximately 325 million labeled traffic flows. Since our method converts network traffic flow into an image, we utilize the original PCAP file of the IoT-23 dataset to build the dataset. Although the original dataset consisted of millions of flows, our dataset contained only a tiny portion of the original dataset. From the PCAP files, each bidirectional network flow was separated by SplitCap [79], and the ground truth information for each flow was gathered from the conn.labeled.log file provided in the original dataset. The statistics of the IoT traffic flow dataset utilized in the experiment are presented in Table 4.2. The dataset had the following seven classes: Attack, Heartbeat, C&C, PartOfAHorizontalPortScan, DDoS, Okiru, and Benign.

Evaluation metrics: We computed the number of true positive (TP), true negative (TN), false positive (FP), and false negative(FN). FP indicates the number of normal data incorrectly classified as an attack, while FN represents the number of attack samples incorrectly classified as normal. TP and TN denote the number of normal or attack data that were accurately classified. Based on these values, the evaluation metrics, such as precision, recall, and F1 score are computed as described in Chapter 1.

TABLE 4.3: Evaluation results of few-shot learning-based network traffic detection system. The evaluation metrics for multi-class classification are macro-averaged.

Model	Test class	4-way 5-shot			4-way 10-shot		
		Precision	Recall	F1 score	Precision	Recall	F1 score
Prototypical Networks [12]	Benign	0.7270	0.8230	0.7720	0.7707	0.8640	0.8147
	DDoS	**0.9738**	0.8540	0.9100	**0.9747**	0.8640	0.9058
	Okiru	0.9585	0.9010	0.9289	**0.9935**	0.9200	0.9555
	PortScan	0.8354	0.8700	0.8562	0.8618	0.9300	0.9020
	Average	0.8737	0.8620	0.8668	0.9002	0.8860	0.8945
FGNN [25]	Benign	**0.9185**	**0.9010**	**0.9096**	**0.9409**	**0.9560**	**0.9484**
	DDoS	0.8863	0.8730	0.8796	0.9612	0.9260	0.9435
	Okiru	**0.8821**	**0.9800**	**0.9285**	0.9306	0.9260	0.9283
	PortScan	0.8613	0.7950	0.8268	0.8918	0.9150	0.9033
	Average	0.8871	0.8873	0.8861	0.9313	0.9308	0.9306
Proposed	Benign	0.8293	0.8160	0.8226	0.8870	0.8870	0.8870
	DDoS	0.9624	**0.9730**	**0.9677**	0.9677	**0.9880**	**0.9777**
	Okiru	0.9594	0.9690	0.9642	0.9694	**0.9510**	**0.9601**
	PortScan	**0.8764**	**0.8720**	**0.8742**	**0.9339**	**0.9320**	**0.9329**
	Average	**0.9069**	**0.9075**	**0.9072**	**0.9395**	**0.9395**	**0.9394**

4.5.1 Results

The effectiveness of the model was measured on a dataset consisting of seven types of network traffic, as illustrated in Table 4.2. The dataset was separated into a meta-training set, which included benign traffic and three types of malicious traffic (Attack, Heartbeat, C&C), and a meta-test set, which is made up of benign traffic and the rest of the malicious traffic. Selecting the DDoS, Okiru, and PortScan classes as the meta-test dataset facilitates the examination of a scenario involving a large volume of unseen and unlabeled malware traffic for analysis. Only these three classes in our dataset satisfy the condition. Moreover, it is possible to train the model with any combination of attack classes as long as the attack classes in the meta-train and meta-test dataset do not overlap since the goal of our few-shot model is to investigate how well the meta-trained model generalizes the unseen classes, given a small number of labeled samples of each unseen class.

Both the training and test datasets had four classes, and the few-shot learning problem became a 4-way K-shot classification problem. Since benign traffic is used in both datasets, 1,690 benign samples are used for training, thus making a total of 8,834 and 74,664 samples for meta-train and meta-test datasets, respectively. As mentioned in Section 4.4.1, to train the few-shot model in an episodic manner, we need to sample multiple tasks such that each task consists of a support set and a query set. We did not explicitly split the meta-train dataset as a support set and query set. Instead, the support and query examples for one training task are randomly sampled from the meta-train dataset. For the meta-test dataset, the labeled 2% of the meta-test dataset is regarded as the support test set and the rest of the unlabeled samples are considered as the query test set. At the meta-testing stage, a test task consists of support and query examples are randomly drawn from the support and query test sets.

As our purpose is to use limited labeled samples as much as possible, our model is trained and tested with 5-shot and 10-shot samples, and the experiment results are presented in Table 4.3. During the meta-training, we train a total of 30 episodes, with each episode consisting of 100 tasks. For instance, for 4-way 5-shot classification with one query image per class, we sample a total of $4 \times (5 + 1) = 24$ images for each task based on the formula $N_{\text{way}} \left(N_{\text{shot}} + N_{\text{query}}\right)$. Therefore, for each episode, we sample a total of $24 \times 1000 = 2400$ images from the meta-train dataset. For simplicity, the number of query samples for one task is chosen as one per class, but we can choose any number of query samples per class in one task.

The experimental results indicated the proposed model achieved an F1 score of 0.91 in 5-shot classification and 0.94 in 10-shot classification. The average recall and precision in both 5-shot and 10-shot classification was above 90%. To exhibit the efficiency of the proposed method, we compared it to two baseline methods: Prototypical Networks [58] and FGNN [68]. The former had an F1 score of 0.8945, while the latter had an F1 score of 0.9306 in a 10-shot setting. The proposed approach outperforms both methods in terms of the average F1 score in both 5-shot and 10-shot classification. Comparing our proposed method to FGNN in 10-shot classification, our model performed only slightly better than FGNN in all average evaluation metrics. However, in the 5-shot classification, all average evaluation metrics of our proposed method were approximately 2% higher than those of FGNN. Thus, our proposed method can effectively classify IoT network traffic using a few labeled samples.

Furthermore, to investigate how well the trained model performs on the attack, benign, C&C, and Heartbeat classes, the evaluation results for those classes in the 4-way 5-shot scenario are illustrated in Fig. 4.7. 10% of each test class is labeled, and used as the support set. The rest of the samples are regarded as the query set that is needed to be predicted by the few-shot learner. The overall accuracy is 93%. The model can correctly classify the attack class; however, the classes C&C and Heartbeat incorrectly predict the output label as each other since the Heartbeat class in the IoT-23 can be considered a sub-class of the C&C class.

4.5.2 Discussion

The proposed model visualizes the individual network flow as an image to train the model. Therefore, the data preprocessing includes two steps: the individual network flow separation and the conversion of each flow as an image. Since the size of the PCAP file in the IoT-23 dataset is large and the captured duration is long, generating the bidirectional network flows from the IoT-23 dataset takes longer. However, just a few milliseconds are required to visualize each flow as an image. Making the image dataset (i.e., Table 4.2) from the network flows takes approximately 40 minutes. Since our few-shot model operates upon the graph structure, it takes additional cost to construct the graph. As the graph is built for each few-shot task, for instance, for 4-way 5-shot classification with one query image per class, each task consists of 20 support images and 4 query images, producing 24 nodes for the graph. Since the number

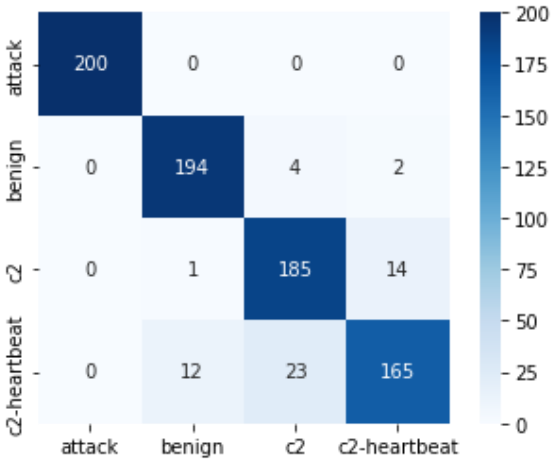

Fig. 4.7. Confusion matrix of few-shot model on test dataset. The 4-way 5-shot few-shot model is trained on attack, benign, C&C, and Heartbeat classes

of nodes is small and the prototype computation module has already estimated the edges, the graph can be generated immediately. The training time of the proposed model for 30 episodes, excluding the data preprocessing time (i.e., network flow separation and image visualization), takes approximately 20 minutes for 4-way 5-shot learning with 100 tasks in one episode.

Even though the proposed model exhibited excellent performance, it also has limitations, such as the validity of the dataset and the deployment of the proposed model in a real environment. The number of IoT devices deployed to create the IoT-23 dataset is somewhat small, and the variations of the simulated attack are considerably fewer, which could further limit the real-world capability of the IoT-23 dataset. Though the IoT-23 dataset used in the study was published in 2020, some IoT traffic was captured in 2018. It means the trained few-shot model is behind the current IoT ecosystem by almost four years. Since IoT is an emerging technology, this time gap enables various changes in IoT devices, such as protocol, firmware, and OS version. Consequently, these changes can open up a potentially exploitable environment for malicious attackers. Therefore, constant supervision and frequent updates of the IoT data are required to maintain the model's effectiveness.

Our insight on the possibility of deploying the proposed model in the real environment is that as long as the individual network flow is provided into the proposed model in real-time, the deployment of the trained model should be possible for real-time detection. The testing time of the proposed model might be inferior to the conventional machine learning model. This is because the model needs to construct the few-shot task from the meta-test dataset and create the graph for traffic classification. However, considering the resource-limited IoT devices, the size of the classifier and the data preprocessing step could be a heavy burden while deploying on a single IoT device. Instead, the proposed model could be deployed in a place (e.g., an edge server) with sufficient resources to analyze the network traffic transmitted by the IoT devices in real-time. One advantage of the proposed few-shot learning over conventional machine learning

is that the model retraining is unnecessary if a new type of attack has emerged. We only need to add a limited labeled sample of the unseen attack classes in the support set of the meta-test dataset so that the trained model can classify the unseen classes.

4.6 Conclusion

In this study, we propose a few-shot meta-learning approach based on the prototypical graph neural network to classify malicious network traffic in IoT devices. Although ML models perform reasonably well in recognizing malicious samples, they require a considerable amount of traffic samples to train the model. Our proposed method addresses this issue by utilizing an episodic few-shot learning paradigm. We transform the bidirectional network flows into images, and a pre-trained CNN model can automatically extract useful features from network packet data. Our model is designed to learn how close the two network flows are in the embedding space, by utilizing the Euclidean distance function. With this information, the model constructs the traffic image as a graph structure and classifies it using the proposed approach.

The experimental results demonstrated that the proposed model outperformed the baseline models. Our approach identifies malicious IoT traffic with an F1 score of 0.94 and 0.91 in 10-shot and 5-shot classification, respectively. Even though the advantage of few-shot learning is that the training and test classes do not need to be the same, it has limitations. Since the number of classes is fixed during the meta-training, if we train the model with N classes, unfortunately, only N classes can be predicted at a time in the meta-testing stage. Moreover, the IoT malware attack categories studied in this research comprised only a few attack categories. Therefore, this research can be further extended to experiment with the new attacks. Furthermore, the proposed model is simulated in a controlled environment; therefore, we could not exactly elaborate on the practical operational efficiency of our proposal in real-time.

Chapter 5

Personalized Federated Learning-based Intrusion Detection System: Poisoning Attack and Defense

5.1 Abstract

To deal with the increasing number of cyber-attacks, intrusion detection system (IDS) plays an important role in monitoring and ensuring the security of the computer network. With the power of machine learning and deep learning, intelligent IDS systems have gained increasing attention due to their efficiency and high classification accuracy. However, the premise of machine learning/deep learning is that the data must be in one central entity (e.g., server) to train the model. This causes additional concerns, such as data transmission costs and privacy leakage. Federated learning complements this shortcoming with a privacy-preserving decentralized learning technique. In federated learning, the data are not shared with the server, local model training is performed where the data reside and only the model parameters are exchanged with the server. This study investigates the federated learning-based IDS approach in the context of IoT data to tackle the main challenges imposed by federated learning. Data heterogeneity and poisoning attacks launched by malicious clients are the main focus of this study. As real-world IoT datasets are heterogeneous, we propose a personalized federated learning-based IDS approach to handle imbalanced data distributions. Moreover, a curious yet malicious client can poison the local data or model to corrupt the global intrusion detection model due to the distributed nature of federated learning, where the central server has no control over the client's local training process. This study demonstrates that the existence of a malicious client can degrade the performance of the federated learning-based IDS model. Accordingly, we propose a robust approach called pFL-IDS to combat poisoning attacks against the federated learning-enabled IDS on heterogeneous IoT data. In comparison with the baseline methods, we demonstrate that our pFL-IDS can detect poisoning attacks without compromising performance.

5.2 Introduction

With the rapid development of the Internet, the world has become more connected, and the deployment of the Internet of Things (IoT) devices has increased. It is reported that IoT devices will reach 55.7 billion by 2025 [80]. The massive amount of data generated by these

devices, combined with their vulnerable nature, opens more attack interfaces and opportunities for malicious parties to conduct cyber-attacks [81]. In 2016, the infamous Mirai botnet attack triggered Internet security breaches by compromising massive IoT devices [52]. This further signifies the need to strengthen the security of the IoT network ecosystem to protect the network from cyber-attacks. Therefore, an in-depth network traffic analysis is necessary; the intrusion detection system plays a vital role in detecting and mitigating unwanted attacks.

To date, different techniques have been utilized in network intrusion detection systems to detect network anomalies, which are either signature-based or behavior-based approaches or a combination of both [82]. Combined with its efficiency and effectiveness, the machine learning/deep learning-based IDS was proven to be a perfect candidate with high detection accuracy. However, the traditional machine learning-based method requires the data to be in one central place to analyze the data gathered from all user devices. This increases the data transmission costs and the risk of privacy leakage as the data from user devices contain sensitive or confidential information.

To resolve these issues, McMahan et al. [83] proposed a privacy-preserving and distributed federated learning paradigm for on-device learning. Federated learning is a client–server architecture that comprises multiple clients tied to a central server. It enables clients/devices to train a shared model collaboratively without the actual data exchange to guarantee user privacy, reduce data transmission costs, and improve the overall model accuracy. In the context of IoT networks, research on federated learning-based IDS has been emerging recently [82]. In this system, the clients train the global intrusion detection model on their local dataset to compute and upload local model updates to the server. The server combines local models to generate the global intrusion detection model, which is sent back to the clients for further training. This process is repeated until the global model converges.

Despite its benefit, federated learning still has some limitations, such as difficulty in handling the heterogeneity in the data distribution of the client and the poisoning attacks executed by malicious clients [82, 84, 85]. In realistic scenarios, the client data are confirmed non-independent and identically distributed (non-IID) in contrast to the ideally assumed independent and identically distributed (IID) scenarios seen in many research methodologies. In federated learning-based IDS, some clients (devices) may only have attack traffic, while others may have normal traffic only or consist of different kinds of attack traffic. Moreover, as demonstrated in recent studies [86, 87], federated learning-based IDS is prone to data poisoning and model poisoning attacks as the server has no control over the behavior of the client.

Poisoning on data can be done either by modifying the ground truth labels of the client dataset (label-flipping attack) or injecting false training data (clean label attack). In model poisoning, malicious clients can manipulate local model parameters to align the global model's objective closer to the attacker's objective. Both poisonings can cause misjudgment in the IDS model, for instance, recognizing benign traffic as attack traffic or vice versa. To alleviate these issues,

vigorous methodologies have been proposed by the research communities, including the minimization of the influence of the malicious client using robust aggregation techniques in the server [88, 89] and detection and removal of the poisoned clients [90–92] before the global model aggregation.

The objective of our study is to model the robust federated learning-based IDS that can effectively detect network anomalies of IoT data. In line with the ongoing research efforts for federated learning, our proposal explores two topics: the effect of data heterogeneity and the behavior of the federated learning-based IDS when poisoning attacks occur. We examine two poisoning attacks: label-flipping [93] and model update poisoning attacks [94], to understand the impact of the poisoned clients against the federated learning-based IDS. First, we analyze the consequences of these attacks and evaluate how well the existing robust server aggregators [88, 89] can defend against these poisoning attacks. According to our findings, the existing robust aggregators cannot effectively mitigate the influence of the poisoned clients when the client's data heterogeneity is significant.

Therefore, we designed the poisoned client detector on the server that can distinguish the poisoned clients from the non-poisoned clients and restricted their participation from the global model aggregation to achieve our objective of reducing the impact of poisoned clients on the global intrusion detection model. The concept of detecting poisoned clients has been explored in previous studies [90–92]. Nevertheless, existing approaches operate under the assumption that the ratio of poisoned clients is small or they need a clean server dataset to compare the client model with the server model. Otherwise, these approaches are not tailored to work well with non-IID data. On the contrary, our detection approach can identify poisoned clients without drastically degrading the overall performance, and it does not require an additional clean dataset on the server.

In this study, we analyze the behavior of label-flipping attacks and model update poisoning attacks against the federated learning-based IDS for IoT data under non-IID data scenarios. The main contributions of this study are summarized as follows.

1. We propose personalized federated learning for intrusion detection system (pFL-IDS), a robust federated learning-based IDS model with a poisoned client detector, to handle IoT data heterogeneity and poisoning attacks.

2. With the logit adjustment loss, we model the personalized federated learning approach customized to the client's data distribution. We demonstrate that personalized logit adjustment loss is suitable if the data distribution between clients varies considerably, which is a common occurrence in IoT data.

3. To mitigate the influence of the poisoned clients from the global intrusion detection model, we design a poisoned client detector at the server. Our detector can identify poisoned clients and limit their participation in the global intrusion detection model aggregation.

4. We evaluate proposed pFL-IDS with the N-BaIoT dataset [15] to study the effectiveness of our approach. We sample different data scenarios from the dataset to simulate the imbalanced data and experiment with different poisoning attacks.

5. We demonstrate our pFL-IDS achieves better performance than state-of-the-art methods [88, 89, 91] in the detection of IoT network traffic anomalies even if some clients are deliberately poisoned in the federated training process.

5.3 Background

5.3.1 Federated Learning

Federated learning is a distributed machine learning technique that collaboratively trains the learning algorithms on multiple edge devices or clients, assuming that the client's private information does not leave the local machine. Generally, federated learning can be viewed as a client–server architecture that comprises multiple clients tied to a central server. Each client trains the local model using local data and shares the trained model parameters (not data) with the rest of the clients through a central server. The central server combines all local models to create a unique global server model, which is sent back to the client for further training. After several communication rounds, the global model that contains the knowledge of multiple clients is obtained. As federated learning is iteratively trained, the learning model can be updated in each round, improving overall accuracy.

Compared to traditional centralized machine learning, which requires the data to be placed in one central location to train the model, a federated learning framework can train the model without such constraint, subsequently reducing data transmission costs and privacy leakage. Moreover, previous studies have demonstrated that federated learning performance is comparable to centralized learning. Since federated learning can realize data integration and model fusion while developing high-precision learning models, it can be utilized in developing high-quality big data services. As illustrated in Fig. 5.1, the typical procedure for federated learning includes local training, model parameters exchange between clients and the server, and global model aggregation. FedAvg [83] is the mainstream server aggregation algorithm to update the global model by taking a weighted average of the client's model parameters. Besides FedAvg, there are other aggregation methods, such as FedProx [95], FedNova [96], Scaffold [97], MOON [98], and Per-FedAvg [99]. These models are developed to address the communication overhead, data privacy concerns, and data heterogeneity in federated learning.

Based on the client size, federated learning can be categorized into cross-device and cross-silo federated learning. In cross-device federated learning, the client number is typically large (e.g., up to millions), with each client likely to have a relatively small amount of data.

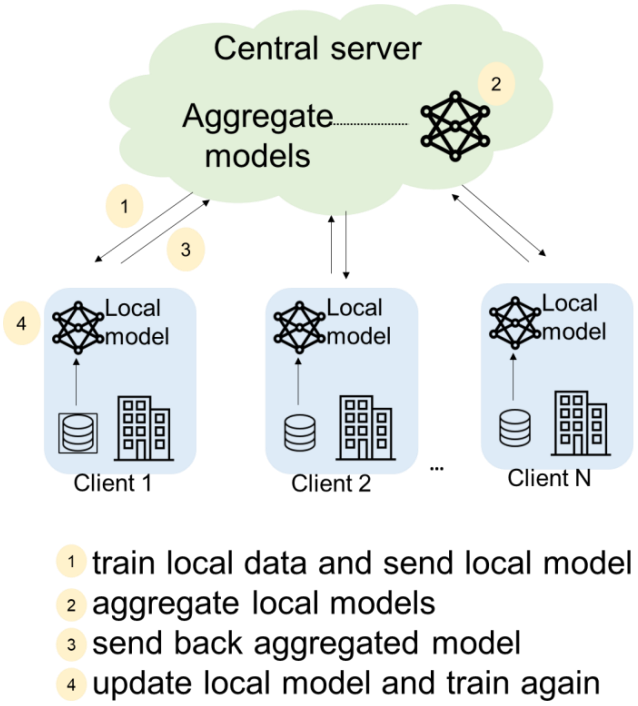

Fig. 5.1. Federated learning system

Examples include smartphones, edge devices, and wearable devices. On the other hand, cross-silo federated learning is generally associated with large organizations or companies where the participants are relatively small (e.g., up to a hundred) and have enough computing power. In the cross-device scenario, the client's participation in every communication round is infeasible due to the large number of clients. Therefore, to avoid bottlenecks, only a few clients can participate in every round. In contrast, every client is expected to participate in every round of the cross-silo federated learning.

Moreover, depending on the type of data available, feature space, and the model exchange method, federated learning can be categorized into horizontal, vertical, and federated transfer learning. Horizontal federated learning is applicable if each client has a different dataset with the same feature spaces. On the other hand, vertical federated learning is suitable if each client has common entities with varying feature spaces. Federated transfer learning is appropriate if each client has different domains or related tasks (e.g., transferring a pre-trained image classification model to video classification). Due to its advantages, federated learning has been applied in various fields, such as healthcare, the financial industry, and IoT networks. In this study, we investigate the application of federated learning to the cybersecurity domain, especially for the intrusion detection system. Since our approach involves a relatively small number of clients, each equipped with different network traffic samples but shared feature spaces, we have utilized the cross-silo and horizontal federated learning scenarios in our proposed model.

TABLE 5.1: A summary of existing works on federated learning-based IDS

Reference	Personalized FL	Defense methods for poisoning attacks	Neural network	Main contributions
Val et al. [86]	-	Coordinate-wise median and trimmed mean	MLP, Autoencoder	Evaluate the resilience of FL models against malicious clients
Popoola et al. [100]	-	-	DNN	Detection of zero-day IoT botnet attack
Fan et al. [101]	✓	-	CNN	Learn customized IDS model using federated transfer learning
Mothukuri et al. [102]	-	-	GRUs with a Random Forest ensembler	Combine the output of different GRUs layers with random forest ensembler
Attota et al. [103]	-	-	ANN	Utilize various data views of IoT traffic to maximize the detection accuracy
Ferrang et al. [84]	-	-	DNN, CNN, and RNN	Comprehensive survey and experimental analysis of FL for cyber security
Ours (pFL-IDS)	✓	Poisoned client detector	CNN	Personalized FL-based IDS for non-IID data with a poisoned client detector

5.3.2 Existing Study on Federated Learning-based IDS

Privacy-preserving federated learning has been extensively applied in the context of network intrusion detection systems due to its reputation of not needing to exchange private data. Val et al. [86] proposed federated learning-based IoT network anomaly detection based on both supervised deep learning and unsupervised autoencoder. Moreover, they considered the presence of adversarial poisoning attacks and evaluated how well the existing robust aggregators [88, 89] can mitigate the impact of the poisoning attacks.

Popoola et al. [100] proposed zero-day botnet attack detection for IoT edge devices and demonstrated that federated learning-based DNN outperformed the conventional DNN model in terms of attack detection accuracy, low communication overhead, and data privacy. Fan et al. [101] proposed IoTDefender, a transfer learning-based IDS for 5G IoT. IoTDefender combined data using federated learning and learned personalized attack detection models by transfer learning while ensuring privacy.

Monthukuri et al. [102] proposed federated learning-based anomaly detection for IoT networks based on GRUs. They improved the classification accuracy by combining the predictions from different layers of GRUs models with an ensemble of the random forest of models, demonstrating the improvement of the attack detection compared to centralized machine learning. Attota et al. [103] proposed MV-FLID, an ensemble multi-view federated learning for IoT intrusion detection. MV-FLID learned three separate ANN models for three views of network traffic (i.e., uniflow, bi-flow, and packet). The predictions of these models were combined through a random forest model to improve the attack detection accuracy.

Ferrag et al. [84] provided a comprehensive survey on federated learning for IoT intrusion detection systems. They demonstrated that federated learning outperformed centralized learning by evaluating three IoT traffic datasets with different deep learning models. The aforementioned

studies, as summarized in Table 5.1, demonstrated the effectiveness of federated learning for intrusion detection; however, most of them did not consider data heterogeneity and how to combat the possible poisoning attacks launched by malicious clients. Therefore, our study examines both issues to improve the federated learning-based IDS model.

5.3.3 Impact of Statistical Heterogeneity in Client Data

One of the significant challenges in implementing federated learning is the presence of statistical heterogeneity in data (non-IID), which is more apparent in the cross-device federated learning scenario. While centralized machine learning techniques assume the data to be independently and identically distributed (IID), federated learning often has an issue with that assumption due to its distributed nature, for example, non-IID data in client data distribution and other statistical factors. This kind of statistical heterogeneity in data can cause client data bias (causing client drift while updating local models), resulting in the performance degradation of the aggregated global model. Generally, data heterogeneity can be classified into quantity skew, label distribution skew, and feature distribution skew. Since our study is focused on label distribution and quantity distribution skews, we only provided a detailed explanation of these two scenarios.

Quantity skew: It refers to the situation where each client has different amount of data samples. In practice, it is almost impossible for every client to have equal amount of data when training the federated learning model. To simulate this scenario, the authors in [104] partitioned the CIFAR-10 dataset with different amounts of data for each client, and experimented with the partitioned datasets. Investigating how the uneven amount of data among clients impacts the model performance is crucial in federated learning.

Label distribution skew: It refers to the variation in the distribution of data labels among the clients. For example, in previous studies [105, 106], the Dirichlet distribution Dir(η) is often utilized to simulate the data distribution differences for evaluating the performance of federated learning. Here, the value of η controls the degree of data heterogeneity. The smaller η value means that the non-IIDness of the data is high [106]. Therefore, by manipulating the η value, we can generate various data distributions to examine the impact of the data heterogeneity on the federated learning model. Furthermore, in studies such as [83, 95], the authors utilized two label categories out of ten to simulate the non-IIDness of the data. For example, in the case of the MNIST digit dataset (labels ranging from 0 to 9), the authors [83, 95] distributed data containing only two digits to each client. Subsequently, each client's data may or may not overlap with other clients. In label distribution skew, the overall data quantity for all clients is often fixed; for instance, the client may have 100 data samples corresponding to each label.

To address the negative influence of the non-IID data on federated learning, model personalization approaches have recently received much attention in research communities. There are

various techniques for model personalization, such as local model fine-tuning, multi-task learning, model clustering, model parameter decoupling, and knowledge distillation [107]. Out of these approaches, our proposed model deals with the non-IID data through a model decoupling strategy. Decoupling means that the neural network model is partitioned into a feature extractor (body) and classifier (head). Then, by applying the mini-batch logit adjustment loss to the head classifier on top of the commonly used cross-entropy loss, which is explained in Section5.4.2, the negative impact of non-IID data can be reduced. With these two losses, each client optimizes its local model based on the importance of the underlying class distribution, making the global model more resilient to the non-IID distribution.

5.3.4 Poisoning Attacks against Federated Learning-based IDS

We assume that the federated server is not compromised and only consider the poisoning attacks that can occur at the client. The number of attackers is assumed to be less than 50% of the total client. We define the attacker's goals and capabilities as follows.

Attacker's goals: The attacker aims to corrupt the global intrusion detection model either by label-flipping or model poisoning such that the global model outputs wrong predictions on the IoT network traffic. For instance, the global intrusion detection model will misjudge malicious IoT traffic as benign or vice versa and reduce the performance of the model.

Attacker's capability: The attacker can modify the labels of the local dataset or the trained model parameters to achieve the desired goals, but it cannot modify the data. This study investigates two kinds of poisoning attacks, such as label-flipping and model update poisoning attacks, to understand the behavior of poisoning attacks on federated learning-based IDS.

Label-flipping Attack: The attackers manipulate the label of the local dataset to inject falsified local model updates into the server aggregator. The attacker flips the selected source class of the training data to the target class without modifying the features. In the federated learning-based IDS system for anomaly detection, depending on the attacker's goal, the label-flipping can be done in three ways as follows.

1. *Flip benign as the attack label*: The goal is to always predict the traffic as an attack such that the model will have a high false positive rate (FPR) (i.e., false alarms)

2. *Flip attack as the benign label*: As the goal is to always predict the traffic as benign, the model will not correctly detect the attack traffic making the true negative rate (TNR) close to zero.

3. *Flip both labels to each other*: The goal is to make the model's accuracy close to zero.

Model Poisoning Attack: Two types of poisoning are examined by modifying the parameters of the local model: the model update scaling and the same global model attacks.

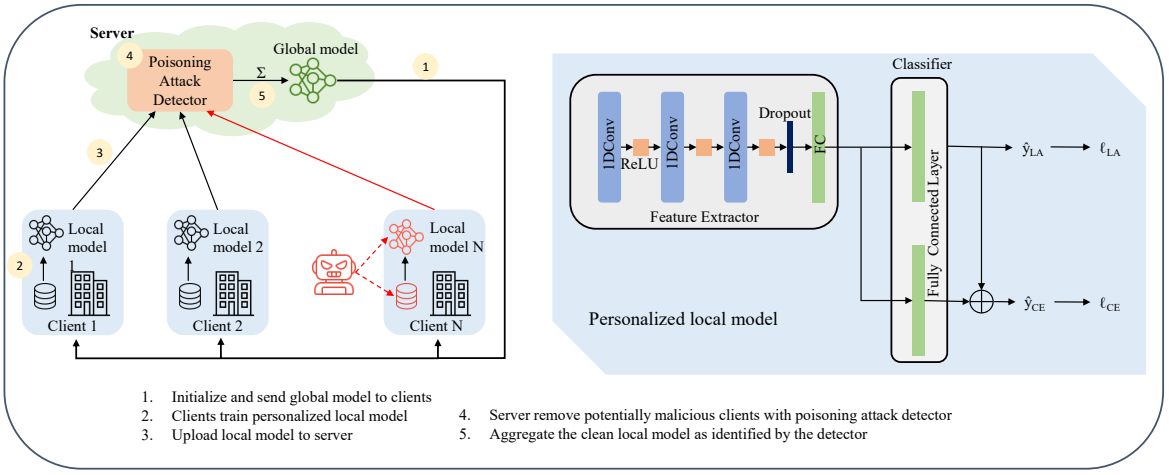

Fig. 5.2. The architecture of the federated learning-baed IDS model

1. *Model update scaling attack*: Malicious clients poison the model by multiplying the local model parameters with the negative scaling factor to corrupt the local model such that the gradient of the poisoned model will be in the opposite direction as that of the benign model. This attack is easy to perform and does not require prior knowledge of client data.

2. *Same global model attack*: Similar to the previous attack, this attack does not train the local model at all but replaces the global model parameters with the same number for all poisoned clients.

5.3.5 Defending Poisoning Attacks in Federated Learning

To defend against the poisoning attacks in federated learning, a variety of robust server aggregation algorithms have been introduced so far, such as coordinate-wise median [88], coordinate-wise trimmed mean [88], and multi-Krum [89]. These methods are designed to deploy on the server in the place of FedAvg and are intended to reduce the negative impact of the poisoning attacks. In addition to these robust aggregators, there is also a method of detecting the poisoned clients before starting the global model aggregation process. The intuition behind this approach is that as the objective of the poisoned clients is different from the non-poisoned clients, their gradients should be close to each other but far away from the rest of the non-poisoned clients. Both techniques provide protection against poisoning attacks to some extent.

Existing studies on poisoning attacks and defense mechanisms in federated learning are mainly centered around the image domain, utilizing datasets like CIFAR-10 and MNIST image datasets. Very few research methodologies have been proposed so far in the cybersecurity domain, particularly for IDS systems. Val et al. [86] investigated data and model poisoning attacks against the federated learning-based IoT network anomaly detection system and evaluated how well the existing robust aggregators [88, 89] can mitigate the impact of the poisoning attacks. Zhang et

al. [87] proposed a robust federated learning-based IDS model by introducing a model-level defense mechanism. Their defense mechanism is based on the online unsupervised poisoned model detection on low-level model parameter representations. Most of the previous studies on the federated learning-enabled IDS approach did not investigate resilience to both data poisoning and model poisoning attacks, especially under IoT data heterogeneity. Therefore, this study delved into those unexplored aspects to provide valuable insights into the significant challenges and novel findings. The following defense methods were implemented in the experiment to compare and evaluate the effectiveness of our proposed defense mechanism.

Coordinate-wise median: It sorts the i^{th} parameters of n local models and takes the median of the i^{th} parameters, which is $w^i = \text{median}\left\{w_n^i : n\varepsilon N\right\}$. When n is an even number, the mean of the two middle values is the median value. When n is odd, the median is the middle parameter.

Coordinate-wise trimmed mean: Similar to the coordinate-wise median, it sorts the i^{th} parameters of n local models and removes the smallest and largest parameters β before the computation of the mean of the rest of parameters $n - 2\beta$.

Multi-Krum: It is a variant of Krum [89], which computes the Euclidean distance between model parameters and averages the top $k = n(\text{total clients}) - f(\text{poisoned client}) - 2$ nearest neighbors to obtain the global model. If $k = 1$, multi-Krum becomes Krum. If $k = n$, it becomes the same as FedAvg. The multi-Krum poorly performs on non-IID data since it is designed to work with IID data.

Poisoned Client Detector: The goal is to detect the poisoned clients and limit their participation in the global model aggregation such that the attacker's goals cannot be achieved. The previous work by Jebreel et al. [91] detects the poisoned client by calculating the cosine similarity of the dimensional-reduced last-layer gradients of local models from the non-poisoned centroid. They assumed that the number of poisoned clients is not larger than 20% so that the normal centroid is always located in the range of the non-poisoned clients, and hence, the poisoned clients can be efficiently identified and removed. As a result, the model performance may be degraded if the poisoned client is larger than the assumed 20%. Moreover, even though the authors have experimented with non-IID data distribution, their proposal did not investigate how to deal with the non-IID data without compromising performance. Compared to [91], our pFL-IDS can defend against poisoned clients up to 30% with a low attack success rate. Moreover, our model is designed to handle performance degradation caused by non-IID clients. The detail of our server-side poisoned client detector is presented in Section 5.4.3.

5.4 Proposed Method

The overview of the novel personalized federated learning approach for the intrusion detection system that is designed to work with non-IID IoT data is described in this section. Considering the vulnerabilities of federated learning against poisoning attacks, we propose a poisoned client detector on the server before proceeding with the usual global model aggregation step of federated learning. Fig. 5.2 shows the architecture of our pFL-IDS. The federated learning process is as follows.

1. **Global model initialization**: At the start of the communication round, a global intrusion model is initialized at the server and is sent to the clients.

2. **Local training**: Each client trains the global model with private data to produce the local model.

3. **Server Aggregation**: The clients upload their local models to the server, which updates the global model parameters with the aggregation of the client model parameters. The aggregator FedAvg [83] is a simple weighted average of the client's model parameters as defined by $w^{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_k^t$. Our model has an additional step before the global model aggregation to mitigate the negative impact of the poisoning attacks on federated learning. We identify the possible poisoned clients and prohibit them from joining the global aggregation process. Our poisoned detector first analyzes the last layer of all client models to determine the poisoned model updates. The proposed detector is based on two-step client similarity alignment and re-weighting. In the first step, we take the pre-computed global model by averaging all local models. As all poisoned clients have the same objective, the poisoned model should be closer to the pre-computed global model if the poisoning attempt is successful. Considering this, we can identify a potential non-poisoned client. In the second step, we compute the angular deviation between local model updates from the dimensionality-reduced centroid of the non-poisoned client. The non-poisoned client will have a smaller angular deviation from the centroid compared to the poisoned client. With that angular similarity, we re-weight each model and take the average of the potentially clean models as a global model for the next round.

4. **Model transmission**: The updated global model is sent back to the clients for further training.

Steps 2 to 4 is repeated for multiple communication rounds until a superior global intrusion detection model is achieved.

5.4.1 Convolutional Neural Network

The remarkable performance and strong feature extraction ability of the two-dimensional CNN have been demonstrated in image classification and computer vision areas. Meanwhile, the

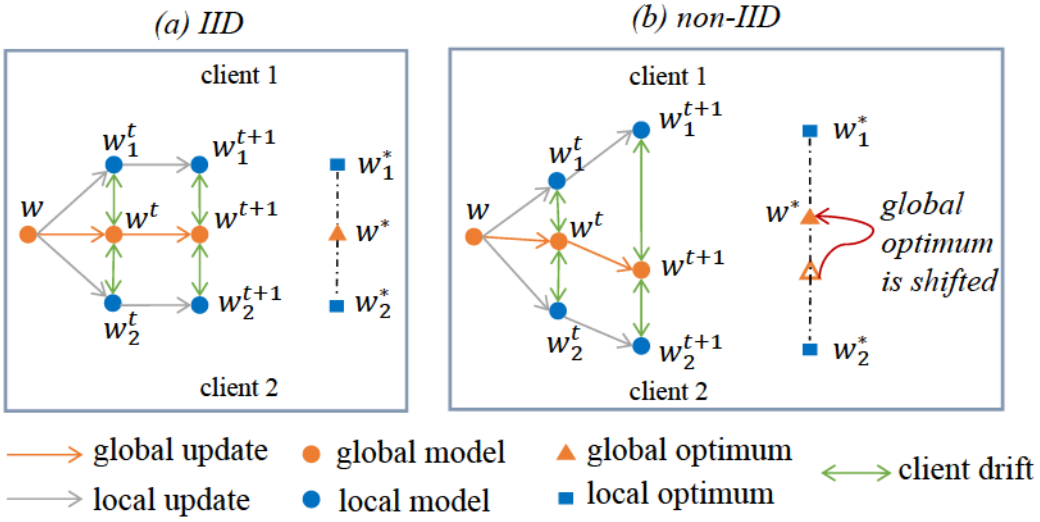

Fig. 5.3. Illustration of client drift in FedAvg algorithm for two clients with 2 local update steps. (a) IID data. The averaged global optimum w^* is located equally from both clients' local optima w_1^* and w_2^*. (b) non-IID data. The averaged global optimum w^* is not equidistant from both local optima due to client drift caused by client imbalanced data distribution. As a result, the global optimum w^* is deviated from the true global optimum, and the global model cannot converge.

one-dimensional neural network (1D-CNN) is designed to work with sequential data, such as audio signals, text, or time-series data. We employ 1D-CNN for the client's intrusion detection model training due to its good performance on time series data. Fig. 5.2 shows our 1D-CNN architecture. It consists of three convolution layers with kernel size 1 and stride 1, followed by the ReLU activation layer. The input of the first convolution layer is 115, which is the total number of features of the N-BaIoT dataset used in the experiment. The convolution layers have 256, 128, and 64 kernel filters, respectively. A dropout layer with $p = 0.2$ is added to prevent overfitting. After the convolution layers map the input to the feature embedding, two fully connected layers are stacked on top to output the correct predictions for the client intrusion detection model.

5.4.2 Local Training: A Personalized Local Model

As the federated learning process involves aggregating multiple local models to produce a single global model, if the imbalanced data distribution is observed among clients, performance degradation happens in FedAvg aggregation. This phenomenon is illustrated in Fig. 5.3. Since the global model is the weighted average of the client's local models, the aggregated global model moves toward the average of the clients' optima. When the clients have IID data, the global model w^{t+1} is closer to the true global optimum w^* as it is the average of the client optimum w_1^* and w_2^*. Therefore, the global optimum w^* is located equally from the local optima, and the client drift did not happen. However, if the clients have non-IID data, the global model w^{t+1} cannot be equally located from both local models since imbalanced data causes client drift when local models are updated (the green line in Fig. 5.3 indicates the degree of

the client drift). Therefore, the global optimum w^* is shifted from the true global optimum, resulting in global model divergence and performance degradation. For example, in Fig. 5.3(b), w^* is closer to client1 optimum w_1^* instead of locating at the true global optimum.

The common approaches for learning class-imbalanced data in centralized machine learning include re-weighting [108–110], which re-weights each class by applying a factor calculated from the number of samples belonging to each class to make a customized loss function and re-sampling [111, 112], which resamples the samples in mini-batch stochastic gradient descent. In this study, according to the idea proposed by [109], we apply the logit adjustment loss function in a mini-batch to balance the client data distribution during local model training.

Logit adjustment loss is a modified version of cross-entropy loss that manipulates the predicted logits to calibrate the unbalanced data quantity of different classes in the logit space. In federated learning, non-IID clients can degrade the performance of the global model, and the logit adjustment loss applied in this work is intended to alleviate those adverse effects by making all clients learn every class accurately. This results in more consistent local models among clients, and the generated global model is more robust to the model divergence and performance degradation. We regard this process as model personalization in federated learning.

Personalized federated learning has been introduced recently to learn a personalized model tailored to each client's data. Extensive efforts have been made to realize a personalized approach for non-IID data in federated learning, including techniques such as multi-task learning, model clustering, model parameter decoupling, and knowledge distillation [107, 113]. [114–117] used the model decoupling approach to learn the personalized model. Similarly, our approach utilizes a local model decoupling strategy for personalized federated learning. We decouple the neural network model into two modules: feature extractor (body) and classifier (head). In this study, our decoupling model includes one feature extractor and two classifiers optimized with two different loss functions. This kind of model decoupling is proven to be effective for both IID and non-IID data since it can optimize both objectives with two different loss functions [117]. In FedAvg [83], the entire network is optimized by the cross-entropy loss function given by

$$\ell_{\mathrm{CE}}\left(y, f\left(x\right)\right) = -\log\frac{\exp(f_y\left(x\right))}{\sum_{y'\varepsilon\mathbb{C}}\exp(f_{y'}\left(x\right))}, \tag{5.1}$$

where $\mathbb{C}$ is the label space and $f_{y'}\left(x\right)$ is the output logits for class y'. In this study, we introduce the additional loss function called mini-batch logit adjustment loss besides the cross-entropy loss to optimize each client's local objective and learn the personalized model for each client, which is defined by

$$\ell_{\mathrm{LA}}\left(y, f\left(x\right)\right) = -\log\frac{\exp(f_y\left(x\right) + \tau.\log\alpha_y)}{\sum_{y'\varepsilon\mathbb{C}}\exp(f_{y'}\left(x\right) + \tau.\log\alpha_{y'})}, \tag{5.2}$$

TABLE 5.2: Statistics of the mini-N-BaIoT dataset

Type		Number of samples
Mirai	Scan	7,000
	UDP	7,000
	UDPplain	7,000
	Syn	7,000
	Ack	7,000
BASHLITE	Scan	9,000
	Junk	9,000
	UDP	9,000
	TCP	9,000
	Combo	9,000
Benign		90,000

TABLE 5.3: Hyperparameters

Neural network parameters	
Local model	1DCNN
Number of conv layer	3
Kernel filters in each conv layer	256, 128, and 64
Number of fully connected layer	2
Number of units in each FC layer	32, 2
Activation function	ReLU
Dropout	0.2
Optimizer	SGD
Learning rate	0.001
Momentum	0.9
Batch size	64
Federated learning parameters	
Number of clients	20
Local training epoch	4
Communication round	50

where, τ is the temperature scaling parameter to re-weight the logit adjustment loss. For simplicity, we set τ as 1. The model learns the feature embedding, followed by the first classifier (head), which is optimized with the logit adjustment loss. We compute the logit-adjusted loss in a mini-batch since it is proven to be effective in handling imbalanced data [118]. After optimizing the first classifier, the second and first classifiers are optimized again with the cross-entropy loss. With these two loss functions, each client learns the local model based on the importance of the underlying class distribution, making the global model more resilient to the non-IID distribution. In summary, our pFL-IDS optimizes the local model with the loss function $\ell_{\text{LA}} + \ell_{\text{CE}}$.

5.4.3 Server-side Poisoned Client Detector

The poisoning attacks aim to reduce the performance of the intrusion detection model such that the model cannot protect the ICT system. It is important to mitigate the influence of poisoned clients from the global intrusion detection model. Therefore, we have proposed a poisoned client detector at the server to identify potential poisoned clients and have limited their participation in the global intrusion detection model aggregation. The detector first analyzes the last layer of all client's models to determine the poisoned model updates. According to [90, 91], the last-layer parameters of the poisoned model behave differently from the non-poisoned model. Therefore, comparing the model's last layer provides sufficient information to distinguish the poisoned models.

Since we aim to restrict the influence of the poisoned clients from the global model aggregation, it is necessary to identify them first before the aggregation. Therefore, we designed the poisoned client detector with the two-step client similarity alignment, followed by model re-weighting. The first similarity computation predicts the potentially normal clients. Then, the normal centroid of the client model is computed from that prediction, and the cosine similarity between

Algorithm 2 Personalized federated learning-based IDS with poisoned client detection

Input: Local dataset $\{D_1, D_2, ..., D_k\}$, Clients $k\varepsilon K_t = \{1, 2, ..., k\}$, Initial global model $w^{t-1}(w^0)$
Output: Global model
Server executes:

for each round $t = \{1, 2, ..., T\}$ **do**
  for every client $k\varepsilon K_t$ **do**
    $w_k^t \leftarrow$ **Client Update**(k, w^{t-1}) ▷ local model
  end for
end for
$\{\delta_k | k\varepsilon K_t\} \leftarrow$ **Poison Detector** $\left(w_k^t | k\varepsilon K_t, w^{t-1}\right)$ ▷ compute the weighted similarity value for clients

$w^{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_k^t * \delta_k$ ▷ re-weight and combine the client models to produce global model

Client Update *(k, w)*:

$B \leftarrow$ split local dataset into batches of size $|B|$
for local epoch i from 1 to E **do**
  for batch $b\varepsilon B$ **do**
    $w \leftarrow w - \eta\Delta\ell\ (w; b)$
  end for
end for
return w to server

Poison Detector $\left(w_k^t | k\varepsilon K_t, w^{t-1}\right)$:

$w_{\text{pre}}^t \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_k^t$ ▷ pre-computed global model

Let $\Delta w_k^t, \Delta w^{t-1}$, and Δw_{pre}^t be the parameters of the last layer of $\left\{w_k^t | k\varepsilon K_t\right\}$, w^{t-1}, and w_{pre}^t, respectively
$\left\{\bar{\delta_1}, \bar{\delta_2}, ..., \bar{\delta_k}\right\} \varepsilon \delta_k^1 \leftarrow cos\left(\Delta w_{\text{pre}}^t, \Delta w^{t-1} - \Delta w_k^t\right)$ ▷ cosine similarity of pre-computed global model and local model

$\left\{\Delta w_{\text{nc}}^h\right\}_{h=1}^{H} \leftarrow \delta_k^1 < 0$ ▷ identify the normal updates

$\left\{\delta_1^*, \delta_2^*, ..., \delta_k^*\right\} \varepsilon \delta_k^2 \leftarrow cos\left(\text{median}\left(\left\{\Delta \hat{w}_{\text{nc}}^h\right\}_{h=1}^{H}\right), \Delta \hat{w}^{t-1} - \Delta \hat{w}_k^t\right)$ ▷ calculate cosine similarity

$\delta_k = -\delta_k^2 + \delta_k^1$ ▷ combine two similarity vectors
$\delta_k = 0$, if $\delta_k < 0$ ▷ normalize the similarity score to [0,1]
$\delta_k = 1$, otherwise
return $\{\delta_k | k\varepsilon K_t\}$

the client models and the forecasted normal centroid is calculated again. After that, these two similarity scores are aligned and normalized into the range of [0,1]. Finally, the client models are re-weighted by the normalized scores during the global model aggregation. First of all, as soon as clients upload the local models, the poison detector at the server calculates the pre-computed global model w^t_{pre}, which is the weighted average of all local models as

$$w^t_{\text{pre}} = \sum_{k=1}^{K} \frac{n_k}{n} w^t_k, k\varepsilon K_t, \tag{5.3}$$

where, w^t_k is the local model of client k and K_t is the total number of clients. After that, we calculate the cosine similarity between the pre-global model and local models as

$$\left\{\bar{\delta_1}, \bar{\delta_2}, ..., \bar{\delta_k}\right\} = cos\left(\Delta w^t_{\text{pre}}, \Delta w^{t-1} - \Delta w^t_k\right), k\varepsilon K_t, \tag{5.4}$$

where Δ indicates the last layer parameters of the neural networks model and w^{t-1} is the previous round global model. The intuition is that the poisoned model should be similar to the pre-global model if the poisoning attempt is successful, as all poisoned clients have the same objective throughout the learning process. Since the cosine similarity value is in the range of [-1,1], according to the above assumption, the similarity values of poisoned clients will be closer to 1 than normal clients. After putting a threshold on the similarity values, we separate the clients into poisoned and normal groups. If the client similarity value is less than 0, we define that client as normal, given by the equation

$$\left\{w^h_{\text{nc}}\right\}_{h=1}^{H} = \delta^1_k < 0, \tag{5.5}$$

where $\delta^1_k \varepsilon \left\{\bar{\delta_1}, \bar{\delta_2}, ..., \bar{\delta_k}\right\}$ and H is the number of normal clients. The median $\left(\left\{\Delta\hat{w}^h_{\text{nc}}\right\}_{h=1}^{H}\right)$ is the dimension-reduced centroid of normal clients. After that, the angular deviations of local models from the normal centroid are computed again by

$$\left\{\delta^*_1, \delta^*_2, ..., \delta^*_k\right\} = cos\left(\text{median}\left(\left\{\Delta\hat{w}^h_{\text{nc}}\right\}_{h=1}^{H}\right), \Delta\hat{w}^{t-1} - \Delta\hat{w}^t_k\right), \tag{5.6}$$

where $\delta^2_k \varepsilon \left\{\delta^*_1, \delta^*_2, ..., \delta^*_k\right\}$ and $\hat{Hat}$ sign refers to the dimension-reduced parameters. The normal client will have a smaller angular deviation from the centroid than the poisoned client. We aligned two similarities as $\delta_k = -\delta^1_k + \delta^2_k$ and normalized it into the range of [0,1] as follows:

$$\begin{aligned} \delta_k &= 0, \text{if } \ \delta_k < 0, \\ \delta_k &= 1, \text{otherwise.} \end{aligned} \tag{5.7}$$

Finally, the local models are reweighted by the normalized similarity scores. The global model for the next round is computed by

$$w^{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} w_k^t * \delta_k, k \varepsilon K_t. \tag{5.8}$$

The detailed procedure of our proposed model is shown in Algorithm 2.

5.5 Evaluation

We evaluated the performance of our pFL-IDS with the N-BaIoT dataset and compared it with several state-of-the-art approaches to investigate the effectiveness of the proposed method. We evaluated the proposed model from the client side. All the experiments are conducted with the PyTorch framework and executed on the PC with Intel Core i7-10750H CPU @ 2.60 GHz, 64 GB RAM.

Dataset: The experiment was performed on the publicly available N-BaIoT dataset [15], which is a collection of network traffic from 9 commercial IoT devices infected with Mirai and BASHLITE. The dataset has 115 features and more than 70 million traffic samples. The dataset has two traffic categories for binary classification (i.e., benign and attack) and 10 sub-categories of attack carried by Mirai and BASHLITE. Since the original dataset has millions of data records, training such a large dataset requires expensive computational resources and higher time complexity. Therefore, we construct a small dataset from the 11 classes of the 9 IoT devices from the original dataset. We take 1,000 traffic records per attack class from each device to build the mini-size N-BaIoT dataset. For example, for the benign class, we select 10,000 traffic data in each device. After combining all the selected traffic records, a mini dataset of 170,000 traffic samples is obtained, as shown in Table 5.2.

We partitioned the mini-N-BaIoT in the ratio of 70 to 30 as the training and test datasets. The training dataset was distributed among the number of participating clients, and the test dataset was used to measure the performance of the federated intrusion detection model. In the practical cross-silo federated learning scenario, the number of clients could be up to a hundred, and to simulate this situation, we further partitioned the training dataset according to the total number of clients. We set the total number of clients used in the experiment as 20. This way of dividing a dataset to mimic the multiple clients has been a well-known approach in the federated learning literature. The hyperparameters used in the experiment are shown in Table 5.3.

Data distribution: The training dataset is further partitioned for N clients to mimic the setting of federated learning. Depending on the data distribution mode (i.e., IID or non-IID), we sampled the client's local dataset from the training dataset. The total number of clients in the

experiment is 20, and we assumed that all clients participated in local training at every round. The following data distribution scenarios are conducted in the experiment.

1. Non-IID: Dirichlet distribution with $\eta = 0.1$
2. Non-IID: Half of the clients have only benign samples and the rest have only attack samples
3. IID: Benign 50%, Attack 50%

In the case of IID, the number of benign and attack samples in the dataset is the same. Non-IID scenario 1 is simulated by the Dirichlet distribution with $\eta = 0.1$. In non-IID scenario 2, we have omitted the attack samples for half of the clients, since the ratio of benign traffic is much larger in the real world and not all IoT devices are compromised. For scenarios 1 and 3, the total number of samples per client is fixed at 1,000. For scenario 2, 500 samples are apportioned to the client that has only benign samples, and 1,000 samples are employed for the rest of the clients.

Poisoning Attack Setting: The behaviors of three label-flipping attacks and two model update poisoning attacks are examined. We assumed that the malicious client is at 30% of the total clients and compared the performance of our pFL-IDS with other baseline methods. Therefore, to give a fair comparison, in the experiment settings of the trimmed mean and multi-Krum, we set the trim ratio β as 0.3 and the poisoned client f as 5. For model update scaling attacks, the gradients are multiplied by a factor of -1 to simulate the attacks. Similarly, all gradients of the poisoned models are replaced by a factor of -3 to mimic the same global model attack.

Evaluation metrics: We compute the number of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) from the experiment results. FP indicates the number of normal data incorrectly classified as an attack, while FN represents the number of attack samples incorrectly classified as normal. TP and TN denote the number of normal or attack data that were accurately classified. Based on these values, the evaluation metrics, such as accuracy, precision, recall, and F1 score are computed. We also calculated the attack success rate (ASR) to measure the accomplishment of the poisoned clients based on their attack objective. If the attack is successful, ASR is 1 (i.e., the global intrusion detection model outputs the target class as desired by the poisoned client instead of the source class); otherwise, it is 0. ASR for label-flipping attacks and model update poisoning attacks are computed as

$$\text{ASR}_{\text{lf}} = \frac{\text{Test}_{\text{target class}}}{\text{Test}_{\text{source class}}}, \text{ASR}_{\text{mp}} = \frac{\text{FP} + \text{FN}}{\text{Total test samples}}. \tag{5.9}$$

Keeping ASR as low as possible while maintaining good performance for the intrusion detection system is necessary for the federated learning-based IDS model to become an effective defense mechanism against poisoning attacks.

TABLE 5.4: Evaluation of non-IID data (scenario 1). The attack ratio is 30%. The best results are bolded.

Poisoning Attack		Metrics	FedAvg	Median	Trimmed mean	Multi-Krum	FL-Defender	Ours
None		Accuracy	**0.9909**	0.9794	0.9859	0.9885	0.9823	0.9822
		F1	**0.9904**	0.9778	0.9849	0.9879	0.9810	0.9809
		ASR	**-**	-	-	-	-	-
Label-flipping	Benign	Accuracy	**0.9857**	0.4706	0.4706	0.4706	0.4706	0.9511
		F1	**0.9850**	0.6400	0.6400	0.6400	0.6400	0.9457
		ASR	0.0267	1.0000	1.0000	1.0000	1.0000	**0.0066**
	Attack	Accuracy	0.9281	0.5294	0.9443	0.7479	0.5294	**0.9613**
		F1	0.9179	0.0000	0.9375	0.6359	0.0000	**0.9575**
		ASR	0.1460	1.0000	0.1123	0.5321	1.0000	**0.0741**
	Both	Accuracy	0.8378	0.4706	**0.9723**	0.4706	0.4706	0.9598
		F1	0.8316	0.6400	**0.9700**	0.6400	0.6400	0.9558
		ASR	0.1622	0.5294	**0.0277**	0.5294	0.5294	0.0402
Model Scaling Attack		Accuracy	0.4706	0.4706	0.4706	0.4706	**0.9776**	0.9642
		F1	0.6400	0.6400	0.6400	0.6400	**0.9759**	0.9607
		ASR	0.5294	0.5294	0.5294	0.5294	**0.0224**	0.0358
Same Model attack		Accuracy	0.4706	0.7367	0.9537	**0.9902**	0.5294	0.9651
		F1	0.6400	0.6117	0.9531	**0.9896**	0.0000	0.9618
		ASR	0.5294	0.2633	0.0463	**0.0098**	0.4706	0.0349

TABLE 5.5: Evaluation of Non-IID data (scenario 2). The attack ratio is 30%. The best results are bolded.

Poisoning Attack		Metrics	FedAvg	Median	Trimmed mean	Multi-Krum	FL-Defender	Ours
None		Accuracy	0.9786	0.7445	0.7807	0.7782	0.5294	**0.9871**
		F1	0.9771	0.6275	0.6979	0.6934	0.0000	**0.9864**
		ASR	-	-	-	-	-	**-**
Label-flipping	Benign	Accuracy	0.9567	0.8544	0.9658	0.8646	0.8190	**0.9891**
		F1	0.9560	0.8184	0.9628	0.8352	0.7627	**0.9885**
		ASR	0.0811	0.0060	0.0110	0.0146	**0.0024**	0.0131
	Attack	Accuracy	0.7610	0.5294	0.5294	0.6369	0.5294	**0.9885**
		F1	0.6609	0.0000	0.0000	0.3749	0.0000	**0.9878**
		ASR	0.5051	1.0000	1.0000	0.7686	1.0000	**0.0098**
	Both	Accuracy	0.7743	0.6405	0.7075	0.6810	0.9721	**0.9893**
		F1	0.6883	0.3850	0.5510	0.4905	0.9697	**0.9886**
		ASR	0.2257	0.3595	0.2925	0.3190	0.0279	**0.0107**
Model Scaling Attack		Accuracy	0.6715	0.5294	0.5297	0.7046	0.9549	**0.9915**
		F1	0.4667	0.0000	0.0011	0.5454	0.9501	**0.9910**
		ASR	0.3285	0.4706	0.4703	0.2954	0.0451	**0.0085**
Same Model attack		Accuracy	0.5294	0.6639	0.8249	0.8605	0.5294	**0.9842**
		F1	0.0000	0.4465	0.7724	0.8284	0.0000	**0.9833**
		ASR	0.4706	0.3361	0.1751	0.1395	0.4706	**0.0158**

TABLE 5.6: Evaluation of IID data. The attack ratio is 30%. The best results are bolded.

Poisoning Attack		Metrics	FedAvg	Median	Trimmed Mean	Multi-krum	FL-Defender	Ours
None		Accuracy	**0.9904**	0.9815	0.9870	0.9896	0.9847	0.9894
		F1	**0.9898**	0.9802	0.9861	0.9890	0.9838	0.9888
		ASR	**-**	-	-	-	-	-
Label-flipping	Benign	Accuracy	0.9892	0.9895	0.9852	0.9886	0.9881	**0.9904**
		F1	0.9886	0.9888	0.9844	0.9879	0.9874	**0.9899**
		ASR	0.0203	0.0105	0.0201	**0.0094**	0.0097	0.0111
	Attack	Accuracy	0.9329	0.8515	0.9685	0.9876	**0.9893**	0.9890
		F1	0.9235	0.8135	0.9657	0.9869	**0.9886**	0.9883
		ASR	0.1385	0.3120	0.0600	0.0113	0.0145	**0.0099**
	Both	Accuracy	0.9271	0.9802	0.9786	**0.9887**	0.9886	0.9877
		F1	0.9164	0.9787	0.9769	**0.9880**	0.9878	0.9871
		ASR	0.0729	0.0198	0.0214	**0.0113**	0.0114	0.0123
Model Scaling Attack		Accuracy	0.9453	0.9628	0.9662	0.9877	0.9876	**0.9885**
		F1	0.9386	0.9592	0.9630	0.9870	0.9868	**0.9879**
		ASR	0.0547	0.0372	0.0338	0.0123	0.0124	**0.0115**
Same Model attack		Accuracy	0.4706	0.9831	0.9879	0.9894	0.9894	**0.9897**
		F1	0.6400	0.9819	0.9871	0.9887	0.9887	**0.9891**
		ASR	0.5294	0.0169	0.0121	0.0106	0.0106	**0.0103**

5.5.1 Results

To demonstrate the effectiveness of our proposal against the poisoning attacks on imbalanced data, the experiment was performed on three data distribution scenarios, and the results are presented in Tables 5.4, 5.5, and 5.6. Moreover, we have compared our pFL-IDS with FedAvg, coordinate-wise median, trimmed mean, multi-Krum, and FL-Detector to study the effectiveness of each model. In addition, the behavior of different kinds of poisoning attacks on non-IID data was also investigated. When the data are IID, most models performed well except the FedAvg, which has the worst performance in the same model poisoning attack scenario. Multi-Krum, FL-Defender, and our pFL-IDS can defend against all poisoning attacks and achieve comparable performance in all evaluation metrics for the IID data. The experimental results for IID data are presented in Table 5.6.

The behavior of poisoning attacks on non-IID data (scenario 1): We sample the non-IID dataset using Dirichlet distribution with $\eta = 0.1$. We choose $\eta = 0.1$ to simulate an extremely imbalanced dataset. The Dirichlet distribution is commonly used in previous studies [90, 91] to sample the non-IID distribution for federated learning. When there is no attack, all models achieve remarkable accuracy and F1 score. However, when facing the non-IID data scenario, our pFL-IDS experiences an approximate 1% accuracy drop compared to the IID data scenario. As shown in Table 5.4, the proposed model has excellent performance and a low ASR value while defending all poisoning attacks. The trimmed mean, and FedAvg are the second-best models, which can defend 3 out of 5 poisoning attacks. The FL-Defender, multi-Krum, trimmed mean and coordinate-wise median cannot absolutely defend against the benign label-flipping attack, with an ASR of 1 indicating that the attacker's goal is 100% achieved. Similarly, a 100% attack

success rate is also observed in the attack label-flipping against the FL-Defender and coordinate-wise median. For model scaling attacks, FedAvg, median, trimmed mean, and multi-Krum have an F1 score of 0.64 with a 0.4706 accuracy. This situation indicates that the intrusion detection model cannot classify the benign label at all (i.e., false alarms will be continuously raised). A similar false alarm situation is also discovered in the same model poisoning attack against the FedAvg model. Meanwhile, our pFL-IDS can defend against both model poisoning attacks with 2-3% accuracy difference from the situation in which no poisoning attacks have occurred.

The behavior of poisoning attacks on non-IID data (scenario 2): In this scenario, the labeled attack samples were absent in half of the clients. Considering not all IoT devices are compromised in real situations, and the labeled attack samples are scarce, the assumption of not having the attack samples in all client's devices is aligned with reality. The empirical results in Table 5.5 indicate that our model performs better than all baseline methods. Even when there are no poisoning attacks, only FedAvg, and our model perform well, while the accuracy of the three robust aggregators drops substantially. FL-Defender has the worst accuracy and F1, with the F1 score being close to zero (i.e., the model cannot classify almost all of the attack samples). This phenomenon is due to having only benign samples in some clients, causing the FL-Defender model to incorrectly identify the clients having both samples as anomalous, subsequently reducing their influence on the global model. Therefore, the resulting global model has little or no knowledge of the attack samples and cannot classify the attack samples. When flipping the label of benign samples as an attack, all baseline models can defend the attack to a certain extent. When attack labels are flipped as benign, besides FedAvg, multi-krum, and our model, the rest of the models have an ASR of 1, implying the goal of the poisoned clients is successfully achieved.

In model update scaling attacks, the attack success rate of all baseline models except the FL-Defender is pretty high, especially 47% ASR and an F1 score close to zero is observed in median and trimmed mean models. This means that the model classified all testing samples as benign, and thus, it cannot detect the intrusion attack samples at all. This situation is the worst as the objective of the intrusion detection system is to classify the attack samples as much as possible while keeping a low false positive rate. For the case of the same model poisoning attack, the aforementioned behavior is observed in FedAvg and FL-Defender. The rest of the baseline models can protect against poisoning to some extent. However, only our methods can effectively protect against all poisoning attacks with excellent performance while preserving superior accuracy and keeping the ASR as low as possible.

5.5.2 Discussion

As the robust aggregators (i.e., median, trimmed mean, multi-Krum) are designed to work with the IID data, they cannot defend against the poisoning attacks when the data are non-IID. As the trimmed mean and multi-Krum must choose the ratio of the malicious attackers, performance

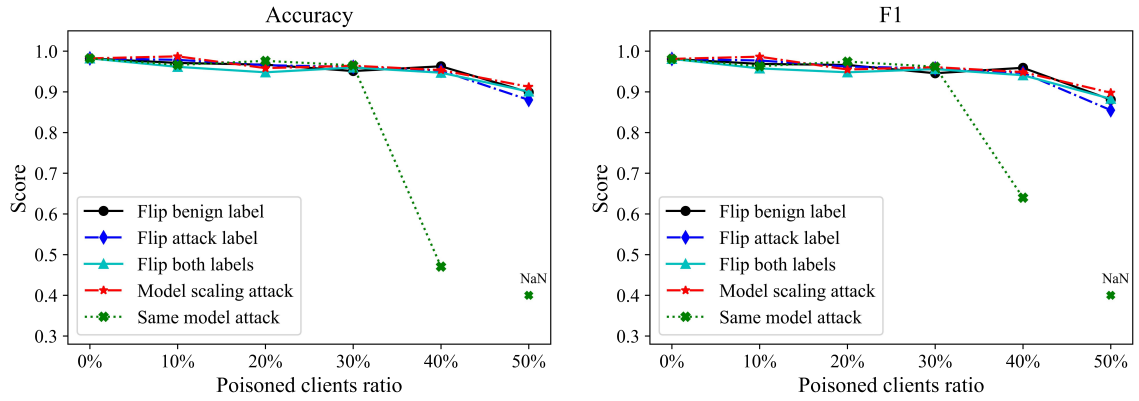

Fig. 5.4. The accuracy and F1 score of pFL-IDS with different ratios of malicious client

degradation could happen if the predefined ratio is less than that of the attackers. The FL-Defender model can correctly detect and migrate the influence of the poisoned clients in the IID mode; however, it cannot protect against all poisoning attacks in non-IID mode, particularly if one class is missing in the client local dataset, as shown in our experimental setting. Due to the client global optima drift phenomenon in non-IID data, the poisoned client detector of FL-Defender may wrongly recognize the benign model as the poisoned model (i.e., caused by client drift). As our pFL-IDS is customized to handle imbalanced data, it works well with non-IID data. The empirical results of all experimented models are presented in Tables 5.4, 5.5, and 5.6.

Impact of different percentages of poisoned clients: The previous studies for poisoning attacks assumed that the portion of the compromised clients is less than 50%, with the poisoned client range of 10%-25% being the most studied one. Accordingly, our study adopted that assumption (<50%). To examine to which extent our pFL-IDS is resilient to poisoning attacks, we first stress-test with a higher ratio of the poisoned clients, and we discovered that pFL-IDS can withstand up to 30% of poisoned clients for all poisoning attacks with a low attack success rate as shown in Tables 5.4, 5.5, and 5.6. The other defense methods collapsed in some poisoning attacks compared to ours, especially apparent in non-IID cases. Even though it is desirable to implement a robust model that can defend as many poisoned clients as possible, having more than 20%-30% of malicious clients in actual practice is rare. Since our model experiments with a few clients (just 20), poisoning 30% of clients required compromising only six clients. However, in large-scale IoT scenarios with millions of clients, even if the attacker wants to poison just a tiny fraction of clients, the attacker needs to compromise multiple clients. This poisoning situation is somewhat impractical and unlikely to happen in the real world. Nevertheless, we investigated the impact of the different ratios of the poisoned clients for label-flipping and model poisoning attacks on our model, and the results are illustrated in Fig. 5.4. When the poisoning ratio reaches 40%, our model cannot defend against the same model poisoning attack despite

being resilient to other poisoning attacks. In principle, our pFL-IDS can protect up to 30% of the poisoning attacks while achieving excellent performance for all evaluation metrics.

Support on large-scale IoT devices: Our proposed model is designed for cross-silo federated learning where the client number is usually small, and each client has sufficient computing power. Thus, in the experiment, we set the total number of clients as 20 and allowed every client to participate in one communication round. However, for large-scale IoT devices, this full participation assumption is infeasible as the server needs to communicate with a large number of IoT devices in each round, which can cause significant communication overhead and computation at the server. Especially since the server needs to wait for all clients to finish uploading the local model to generate a global model, if some clients are struggling to complete the model training, the server needs to wait indefinitely. To support large-scale IoT devices with our method, we can apply the client selection procedure to choose a fraction of the total clients to train in one round. This kind of client selection is commonly utilized in the existing literature. Moreover, to deal with the clients who cannot upload the models in the specified period, we can consider dropping those clients to avoid further delay. Currently, both issues (client selection and handling of the straggling clients) are being vigorously studied in the research community, and we also hope to contribute improvement in that direction in our future work.

5.6 Conclusion

In this study, we investigated the behavior of label-flipping attacks and model update poisoning attacks against the federated learning-based IDS for IoT data under different non-IID data scenarios. Based on our research findings, we designed a robust pFL-IDS model to detect poisoning attacks in federated learning-based IDS. We introduced the logit adjustment loss during local model training to handle data heterogeneity. As real-world IoT network data are undoubtedly imbalanced, we proposed a personalized IDS model for local model training. The empirical results indicated that the existence of malicious clients can degrade the IDS model performance, which is even more severe if the clients have non-IID data. Therefore, we designed a poisoned client detector at the server to identify poisoned clients and limit their participation from the global model aggregation. By conducting extensive experiments, we have demonstrated that our pFL-IDS is effective in combating both poisoning attacks regardless of the client's underlying data distribution. Moreover, the empirical results demonstrated our pFL-IDS outperformed all baseline methods on non-IID data scenarios and proved that it could work well with both IID and non-IID data.

In addition to our research findings outlined in this study, we also acknowledge there are certain limitations. The proposed model is designed for cross-silo federated learning, where the client number is usually small, and each client has sufficient computing power. Therefore, the feasibility of this model for large-scale IoT devices may be limited. Moreover, this study only examined two types of model poisoning attacks, and thus, there is a possibility that the

proposed model cannot defend against more sophisticated poisoning attacks. Furthermore, the experiments were conducted in a controlled environment on a single machine rather than the real client-server architecture; therefore, we could not provide insights into the efficiency of our model for practical operations.

Chapter 6

Conclusion of Our Study

6.1 Conclusion

This research focuses on studying data-driven cybersecurity, particularly utilizing AI techniques to analyze cyber data, revolutionizing the realm of cybersecurity. AI-powered cybersecurity is critical in identifying threat patterns, predicting potential breaches, and deciding the countermeasures. Chapter 1 introduces the background and motivation that has led to this research, highlighting the significance of AI in combating cyberattacks. Moreover, the chapter introduces the basics of AI techniques, their role in data-driven cybersecurity, and how AI can be leveraged to analyze security data. The background of the data-driven cyber intelligence framework, which plays a pivotal role in managing cybersecurity risks, is also discussed. This study centers on developing AI techniques for three categories of cybersecurity data – unstructured text of CTI reports, malicious domain, and network intrusion detection system. To this end, the chapter also elaborates a concise introduction of the research findings on those datasets, which are relevant for the improvement of AI-powered cybersecurity.

Chapter 2 offers a threat modeling technique with the neural network model to identify the cyber kill chain stages of the unstructured text of CTI reports at paragraph-level semantics and to extract the IoCs. The extracted IoCs can be utilized as a threat feed to automatically detect potential threats. This Chapter aims to utilize the example techniques sentences from ATTCK for Enterprise as a training dataset, which is a novel way to classify the cyber kill chain stages for the CTI reports. The experimental results outlined that the proposed model can predict the cyber kill chain phase with an average F1 score of 0.67, highlighting the need for further research and development to improve the model's efficiency.

To further enrich the threat feeds, Chapter 3 of the dissertation introduces a malicious domain detection approach by leveraging active DNS traffic data and WHOIS information. Moreover, this Chapter experimentally verified that incorporating semantic features, in addition to the commonly used lexical and DNS-based features, is effective in detecting malicious domains. The experimental results demonstrated that the proposed approach achieved an accuracy of up to 93% with the random forest classifier. In addition, as a future work, we can further extend the current model to recognize each domain as spam, phishing, or command and control.

While threat feeds can be utilized to protect the network from cyber-attacks, additional measures are necessary to defend against newly emerging malware. Therefore, we shift our study scope to network logs analysis to better understand threats, thereby implementing robust counter-measures. Chapter 4 introduces a few-shot learning-based prototypical graph neural network model that can identify malicious IoT traffic. This Chapter aims to identify the new type of malware traffic with a few labeled traffic samples, thereby minimizing the requirement for model retraining and labeling costs. This study uses the CNN model pre-trained on ImageNet to extract network features from the network images, which is transformed through a tool called Binvis. The proposed model has the ability to learn how close the two network flows are in the embedding space through training on various few-shot tasks, thereby generalizing the acquired knowledge to recognize the new unseen attacks. The extensive experiment revealed that malicious IoT traffic could be predicted with an F1 score of 0.94 and 0.91 in 10-shot and 5-shot classification, respectively.

In Chapter 5, we expand our study of network log analysis to the privacy-preserving and distributed learning framework. This Chapter emphasizes tackling the challenges posed by applying federated learning to cybersecurity data, including data heterogeneity and client poisoning attacks. Moreover, this Chapter discusses the impacts of data poisoning and model poisoning attacks with different data partition scenarios (to simulate data heterogeneity) to propose a robust modeling technique accordingly. The comprehensive experiment indicated that the presence of malicious clients can degrade the federated learning-based IDS model performance, which is more severe if the clients have non-IID data. Therefore, this study proposes a poisoned client detector at the server to combat the poisoning attacks and applies personalization to the federated learning to handle non-IID data. Two poisoning attacks, data poisoning and model poisoning, are experimented with different data partition scenarios: IID and non-IID. With the extensive experiment conducted, we demonstrated that our model is effective in defending against both poisoning attacks regardless of the data distribution of the clients, outperforming almost all baseline methods when experimented with non-IID data and in the presence of poisoning attacks. This success is partly due to the proposed poison client detector, which identifies and blocks the poisoned clients from participating in every round of federated training.

6.2 Contributions

This dissertation makes notable key contributions throughout its four chapters. Chapter 2 introduces a novel threat modeling approach to identify the cyber kill chain stages of the unstructured text of CTI reports. It uses the adversaries' techniques from MITRE ATT&CK for Enterprise as training datasets, while the test dataset is composed of CTI reports from various security vendors. This allows a unique way of mapping threat actions from security text to adversaries' techniques, eventually identifying the cyber kill chain stages described in the paragraph level of CTI reports. Moreover, we can extract IoCs from the CTI reports with the

proposed model, which can be utilized as a threat feed to detect potential threats automatically. This study utilizes various word lists across the Internet to extract IoCs. Though the experiment results outline the potential success of mapping CTI reports to adversaries' techniques, which in turn are labeled with cyber kill chain stages, there is room for further research and development to improve the current state of the model.

Chapter 3 explores the use of different types of features to detect malicious domains, aiming to improve the detection capability. This Chapter introduces 3 groups of features: statistical features, DNS-based features, and domain semantic features. Statistical features offer the fundamental way to identify malicious domains, while DNS-based features provide the association of the domains in the domain name system. As a domain semantic feature, this study computes the domain reputation score with the N-gram methods. We observe that the reputation score of benign domains is higher than that of malicious domains, providing critical insights into differentiating between malicious and benign domains. The incorporation of these three groups of features allows a significant improvement in detecting malicious domains.

Chapter 4 of this dissertation explores the analysis of network logs, particularly IoT network intrusion data, to improve the detection of malicious network traffic. This study also explores the use of the CNN model pre-trained on ImageNet to extract network features from the network images, which is transformed through a tool called Binvis. This chapter offers an innovative way of detecting malicious IoT traffic with the few-shot learning model, which aims to identify the new type of malware with a minimum amount of labeled samples. Specifically, we introduce a novel prototypical graph neural network-based few-shot learning model, which is great for learning how close the two network flows are in the embedding space. Moreover, this approach minimizes the requirement for model retraining and expensive labeling costs associated with newly emerged malware. Our study is the first to challenge the application of graph-based few-shot learning in identifying malicious network traffic. Therefore, Chapter 4 presents the significant research findings and the challenges encountered while implementing the few-shot model.

Chapter 5 introduces a novel federated learning-based network intrusion detection system. This Chapter emphasizes handling the key challenges in federated learning – data heterogeneity and client poisoning attacks. Two poisoning attacks, data poisoning and model poisoning, are examined with different data partition scenarios: IID and non-IID. This study proposes a novel poisoned client detector at the server to combat the poisoning attacks and applies personalization to the federated learning to handle non-IID data. Additionally, this Chapter presents a comparison with other defense models, validating the effectiveness of the proposed model in combating poisoning attacks. Furthermore, our findings highlight the significance of model personalization in handling data heterogeneity in federated learning.

6.3 Future work

This section outlines several research areas that can be further extended as future work to enhance the current state of data-driven cybersecurity.

- **Data Quality and Availability**: The issue with incorporating data-driven intelligence in cybersecurity domain is the availability of high-quality data. Gathering high-quality data from diverse sources is quite challenging as the data may contain confidential information. In addition, most of the AI techniques in the cybersecurity research community are trained on publicly available datasets. Most of those datasets are generated in the simulated network environment. Therefore, it might not reflect the actual sophisticated cyber-attacks. Therefore, experiments on real-world cyber datasets are essential to improve the state-of-the-art detection model.

- **Real-time Implementation**: The experiments in this study are conducted offline using public datasets. Therefore, it is crucial to test the methods mentioned in this study in real-time to assess the effectiveness of the AI models.

- **Privacy Concerns**: The collection of large amounts of cyber data raises major privacy and ethical concerns, as data may contain sensitive information. Extracting valuable insights from data while ensuring data privacy is an ongoing area of research in the research community. Techniques such as differential privacy and federated learning outlined in this dissertation offer a way to alleviate data privacy concerns.

- **Model Interpretability**: The use of AI in data-driven cybersecurity may raise challenges such as model interpretability. Being able to understand how AI models make predictions is critical, especially in the cybersecurity domain, where the experts need to decide and validate how much they can trust the predictions made by AI models. Explainable AI offers a way to interpret AI models, and future works should focus on advancing the interpretability of the AI models.

- **Adversarial attacks**: The adversarial attacks aim to degrade the performance of the AI models by injecting false data (adversarial examples) that are designed to fool the models. Therefore, there is an urgent need to develop AI models that are able to withstand adversarial attacks.

References

[1] N. Kaloudi and J. Li, "The ai-based cyber threat landscape: A survey," *ACM Computing Surveys (CSUR)*, vol. 53, no. 1, pp. 1–34, 2020.

[2] I. Firdausi, A. Erwin, A. S. Nugroho, *et al.*, "Analysis of machine learning techniques used in behavior-based malware detection," in *2010 second international conference on advances in computing, control, and telecommunication technologies*, pp. 201–203, IEEE, 2010.

[3] R. S. Rao and A. R. Pais, "Detection of phishing websites using an efficient feature-based machine learning framework," *Neural Computing and applications*, vol. 31, pp. 3851–3873, 2019.

[4] E. G. Dada, J. S. Bassi, H. Chiroma, A. O. Adetunmbi, O. E. Ajibuwa, *et al.*, "Machine learning for email spam filtering: review, approaches and open research problems," *Heliyon*, vol. 5, no. 6, 2019.

[5] P. Jain, "Machine learning versus deep learning for malware detection," 2019.

[6] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi, "Exposure: Finding malicious domains using passive dns analysis.," in *Ndss*, pp. 1–17, 2011.

[7] K. A. Messabi, M. Aldwairi, A. A. Yousif, A. Thoban, and F. Belqasmi, "Malware detection using dns records and domain name features," in *Proceedings of the 2nd International Conference on Future Networks and Distributed Systems*, pp. 1–7, 2018.

[8] D. Chiba, T. Yagi, M. Akiyama, T. Shibahara, T. Yada, T. Mori, and S. Goto, "Domainprofiler: Discovering domain names abused in future," in *2016 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 491–502, IEEE, 2016.

[9] G. Thamilarasu and S. Chawla, "Towards deep-learning-driven intrusion detection for the internet of things," *Sensors*, vol. 19, no. 9, p. 1977, 2019.

[10] I. Alrashdi, A. Alqazzaz, E. Aloufi, R. Alharthi, M. Zohdy, and H. Ming, "Ad-iot: Anomaly detection of iot cyberattacks in smart city using machine learning," in *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 0305–0310, IEEE, 2019.

[11] Z. Zhu and T. Dumitras, "Chainsmith: Automatically learning the semantics of malicious campaigns by mining threat intelligence reports," in *2018 IEEE European symposium on security and privacy (EuroS&P)*, pp. 458–472, IEEE, 2018.

[12] G. Husari, E. Al-Shaer, M. Ahmed, B. Chu, and X. Niu, "Ttpdrill: Automatic and accurate extraction of threat actions from unstructured text of cti sources," in *Proceedings of the 33rd annual computer security applications conference*, pp. 103–115, 2017.

[13] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization.," *ICISSp*, vol. 1, pp. 108–116, 2018.

[14] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 military communications and information systems conference (MilCIS)*, pp. 1–6, IEEE, 2015.

[15] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-baiot—network-based detection of iot botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018.

[16] A. H. Lashkari, A. F. A. Kadir, L. Taheri, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark android malware datasets and classification," in *2018 International Carnahan conference on security technology (ICCST)*, pp. 1–7, IEEE, 2018.

[17] M. S. I. Mamun, M. A. Rathore, A. H. Lashkari, N. Stakhanova, and A. A. Ghorbani, "Detecting malicious urls using lexical analysis," in *Network and System Security: 10th International Conference, NSS 2016, Taipei, Taiwan, September 28-30, 2016, Proceedings 10*, pp. 467–482, Springer, 2016.

[18] I. H. Sarker, "Cyberlearning: Effectiveness analysis of machine learning security modeling to detect cyber-anomalies and multi-attacks," *Internet of Things*, vol. 14, p. 100393, 2021.

[19] D. B. Rawat, R. Doku, and M. Garuba, "Cybersecurity in big data era: From securing big data to data-driven security," *IEEE Transactions on Services Computing*, vol. 14, no. 6, pp. 2055–2072, 2019.

[20] I. H. Sarker, H. Janicke, L. Maglaras, and S. Camtepe, "Data-driven intelligence can revolutionize today's cybersecurity world: A position paper," *arXiv preprint arXiv:2308.05126*, 2023.

[21] E. M. Hutchins, M. J. Cloppert, R. M. Amin, *et al.*, "Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains," *Leading Issues in Information Warfare & Security Research*, vol. 1, no. 1, p. 80, 2011.

[22] S. Caltagirone, A. Pendergast, and C. Betz, "The diamond model of intrusion analysis," *Threat Connect*, vol. 298, no. 0704, pp. 1–61, 2013.

[23] D. Ito, K. Nomura, M. Kamizono, Y. Shiraishi, Y. Takano, M. Mohri, and M. Morii, "Modeling attack activity for integrated analysis of threat information," *IEICE TRANSACTIONS on Information and Systems*, vol. 101, no. 11, pp. 2658–2664, 2018.

[24] O. Levy and Y. Goldberg, "Dependency-based word embeddings," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 302–308, 2014.

[25] "MITRE ATT&CK." `https://attack.mitre.org/`.

[26] "The New General Service List." `http://www.newgeneralservicelist.org`, Accessed on 2019.

[27] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.

[28] "Wikipedia: List of words about computers." `https://simple.wikipedia.org/wiki/List_of_words_about_computers`, Accessed on 2019.

[29] M. Abu Rajab, J. Zarfoss, F. Monrose, and A. Terzis, "A multifaceted approach to understanding the botnet phenomenon," in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, pp. 41–52, 2006.

[30] J. Woodbridge, H. S. Anderson, A. Ahuja, and D. Grant, "Predicting domain generation algorithms with long short-term memory networks," *arXiv preprint arXiv:1611.00791*, 2016.

[31] B. Yu, J. Pan, J. Hu, A. Nascimento, and M. De Cock, "Character level based detection of dga domain names," in *2018 international joint conference on neural networks (IJCNN)*, pp. 1–8, IEEE, 2018.

[32] H. S. Anderson, J. Woodbridge, and B. Filar, "Deepdga: Adversarially-tuned domain generation and detection," in *Proceedings of the 2016 ACM workshop on artificial intelligence and security*, pp. 13–21, 2016.

[33] J. Peck, C. Nie, R. Sivaguru, C. Grumer, F. Olumofin, B. Yu, A. Nascimento, and M. De Cock, "Charbot: A simple and effective method for evading dga classifiers," *IEEE Access*, vol. 7, pp. 91759–91771, 2019.

[34] L. Sidi, A. Nadler, and A. Shabtai, "Maskdga: A black-box evasion technique against dga classifiers and adversarial defenses," *arXiv preprint arXiv:1902.08909*, 2019.

[35] I. Khalil, T. Yu, and B. Guan, "Discovering malicious domains through passive dns data graph analysis," in *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, pp. 663–674, 2016.

[36] Y. Kazato, Y. Nakagawa, and Y. Nakatani, "Improving maliciousness estimation of indicator of compromise using graph convolutional networks," in *2020 IEEE 17th Annual Consumer Communications & Networking Conference (CCNC)*, pp. 1–7, IEEE, 2020.

[37] Y. Zhauniarovich, I. Khalil, T. Yu, and M. Dacier, "A survey on malicious domains detection through dns data analysis," *ACM Computing Surveys (CSUR)*, vol. 51, no. 4, pp. 1–36, 2018.

[38] A. Kountouras, P. Kintis, C. Lever, Y. Chen, Y. Nadji, D. Dagon, M. Antonakakis, and R. Joffe, "Enabling network security through active dns datasets," in *Research in Attacks, Intrusions, and Defenses: 19th International Symposium, RAID 2016, Paris, France, September 19-21, 2016, Proceedings 19*, pp. 188–208, Springer, 2016.

[39] T. Holz, C. Gorecki, K. Rieck, and F. C. Freiling, "Measuring and detecting fast-flux service networks.," in *Ndss*, 2008.

[40] I. Prieto, E. Magaña, D. Morató, and M. Izal, "Botnet detection based on dns records and active probing," in *Proceedings of the International Conference on Security and Cryptography*, pp. 307–316, IEEE, 2011.

[41] S. Hao, N. Feamster, and R. Pandrangi, "Monitoring the initial dns behavior of malicious domains," in *Proceedings of the 2011 ACM SIGCOMM conference on Internet measurement conference*, pp. 269–278, 2011.

[42] S. Yadav, A. K. K. Reddy, A. N. Reddy, and S. Ranjan, "Detecting algorithmically generated malicious domain names," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pp. 48–61, 2010.

[43] S. Yadav, A. K. K. Reddy, A. N. Reddy, and S. Ranjan, "Detecting algorithmically generated domain-flux attacks with dns traffic analysis," *IEEE/Acm Transactions on Networking*, vol. 20, no. 5, pp. 1663–1677, 2012.

[44] Z. Liu, Y. Zeng, P. Zhang, J. Xue, J. Zhang, and J. Liu, "An imbalanced malicious domains detection method based on passive dns traffic analysis," *Security and Communication Networks*, vol. 2018, 2018.

[45] R. Sharifnya and M. Abadi, "A novel reputation system to detect dga-based botnets," in *ICCKE 2013*, pp. 417–423, IEEE, 2013.

[46] "Alexa Top Sites." `https://www.alexa.com/topsites` Accesssed on June, 2020.

[47] "Malware Domain List." `https://www.malwaredomainlist.com` Accesssed on June, 2020.

[48] "Compromised Domain List." `https://zonefiles.io/compromised-domain-list` Accesssed on June, 2020.

[49] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao, "A survey on security and privacy issues in internet-of-things," *IEEE Internet of things Journal*, vol. 4, no. 5, pp. 1250–1258, 2017.

[50] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications," *IEEE internet of things journal*, vol. 4, no. 5, pp. 1125–1142, 2017.

[51] W. H. Hassan *et al.*, "Current research on internet of things (iot) security: A survey," *Computer networks*, vol. 148, pp. 283–294, 2019.

[52] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, *et al.*, "Understanding the mirai botnet," in *26th USENIX security symposium (USENIX Security 17)*, pp. 1093–1110, 2017.

[53] M. Almiani, A. AbuGhazleh, A. Al-Rahayfeh, S. Atiewi, and A. Razaque, "Deep recurrent neural network for iot intrusion detection system," *Simulation Modelling Practice and Theory*, vol. 101, p. 102031, 2020.

[54] J. L. Leevy, T. M. Khoshgoftaar, and J. M. Peterson, "Mitigating class imbalance for iot network intrusion detection: a survey," in *2021 IEEE Seventh International Conference on Big Data Computing Service and Applications (BigDataService)*, pp. 143–148, IEEE, 2021.

[55] L. Bilge and T. Dumitraş, "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*, pp. 833–844, 2012.

[56] O. Vinyals, C. Blundell, T. Lillicrap, D. Wierstra, *et al.*, "Matching networks for one shot learning," *Advances in neural information processing systems*, vol. 29, 2016.

[57] F. Sung, Y. Yang, L. Zhang, T. Xiang, P. H. Torr, and T. M. Hospedales, "Learning to compare: Relation network for few-shot learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1199–1208, 2018.

[58] J. Snell, K. Swersky, and R. Zemel, "Prototypical networks for few-shot learning," *Advances in neural information processing systems*, vol. 30, 2017.

[59] V. Garcia and J. Bruna, "Few-shot learning with graph neural networks," *arXiv preprint arXiv:1711.04043*, 2017.

[60] R. Shire, S. Shiaeles, K. Bendiab, B. Ghita, and N. Kolokotronis, "Malware squid: A novel iot malware traffic analysis framework using convolutional neural network and binary visualisation," in *Internet of Things, Smart Spaces, and Next Generation Networks and Systems: 19th International Conference, NEW2AN 2019, and 12th Conference, ruSMART 2019, St. Petersburg, Russia, August 26–28, 2019, Proceedings 19*, pp. 65–76, Springer, 2019.

[61] G. Conti, E. Dean, M. Sinda, and B. Sangster, "Visual reverse engineering of binary and data files," in *International Workshop on Visualization for Computer Security*, pp. 1–17, Springer, 2008.
[62] W. Liu, X. Liu, X. Di, and H. Qi, "A novel network intrusion detection algorithm based on fast fourier transformation," in *2019 1st international conference on Industrial Artificial Intelligence (IAI)*, pp. 1–6, IEEE, 2019.
[63] F. Hussain, S. G. Abbas, M. Husnain, U. U. Fayyaz, F. Shahzad, and G. A. Shah, "Iot dos and ddos attack detection using resnet," in *2020 IEEE 23rd International Multitopic Conference (INMIC)*, pp. 1–6, IEEE, 2020.
[64] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM computing surveys (csur)*, vol. 53, no. 3, pp. 1–34, 2020.
[65] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.
[66] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.
[67] J. Kim, T. Kim, S. Kim, and C. D. Yoo, "Edge-labeling graph neural network for few-shot learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11–20, 2019.
[68] T. Wei, J. Hou, and R. Feng, "Fuzzy graph neural network for few-shot learning," in *2020 International joint conference on neural networks (IJCNN)*, pp. 1–8, IEEE, 2020.
[69] G. Bendiab, S. Shiaeles, A. Alruban, and N. Kolokotronis, "Iot malware network traffic classification using visual representation and deep learning," in *2020 6th IEEE Conference on Network Softwarization (NetSoft)*, pp. 444–449, IEEE, 2020.
[70] C. Xu, J. Shen, and X. Du, "A method of few-shot network intrusion detection based on meta-learning framework," *IEEE Transactions on Information Forensics and Security*, vol. 15, pp. 3540–3552, 2020.
[71] Y. Yu and N. Bian, "An intrusion detection method using few-shot learning," *IEEE Access*, vol. 8, pp. 49730–49740, 2020.
[72] "Visual analysis of binary files." `http://binvis.io`.
[73] A. Dainotti, A. Pescape, and K. C. Claffy, "Issues and future directions in traffic classification," *IEEE network*, vol. 26, no. 1, pp. 35–40, 2012.
[74] H. Sagan, *Space-filling curves*. Springer Science & Business Media, 2012.
[75] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
[76] J. Deng, "A large-scale hierarchical image database," *Proc. of IEEE Computer Vision and Pattern Recognition, 2009*, 2009.
[77] A. P. M. J. E. Sebastian Garcia, "Iot23: A labeled dataset with malicious and benign iot network traffic," *Zenodo*, 2020.
[78] "An open source network monitoring tool." `https://zeek.org`.
[79] "SplitCap." `https://www.netresec.com/?page=SplitCap`.
[80] "Future of industry ecosystems: Shared data and insights," 2021. `https://blogs.idc.com/2021/01/06/`

future-of-industry-ecosystems-shared-data-and-insights/ Last accessed on 2022-12-18.

[81] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, "Demystifying iot security: an exhaustive survey on iot vulnerabilities and a first empirical look on internet-scale iot exploitations," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2702–2733, 2019.

[82] S. Agrawal, S. Sarkar, O. Aouedi, G. Yenduri, K. Piamrat, M. Alazab, S. Bhattacharya, P. K. R. Maddikunta, and T. R. Gadekallu, "Federated learning for intrusion detection system: Concepts, challenges and future directions," *Computer Communications*, 2022.

[83] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, pp. 1273–1282, PMLR, 2017.

[84] M. A. Ferrag, O. Friha, L. Maglaras, H. Janicke, and L. Shu, "Federated deep learning for cyber security in the internet of things: Concepts, applications, and experimental analysis," *IEEE Access*, vol. 9, pp. 138509–138542, 2021.

[85] V. Mothukuri, R. M. Parizi, S. Pouriyeh, Y. Huang, A. Dehghantanha, and G. Srivastava, "A survey on security and privacy of federated learning," *Future Generation Computer Systems*, vol. 115, pp. 619–640, 2021.

[86] V. Rey, P. M. S. Sánchez, A. H. Celdrán, and G. Bovet, "Federated learning for malware detection in iot devices," *Computer Networks*, vol. 204, p. 108693, 2022.

[87] Z. Zhang, Y. Zhang, D. Guo, L. Yao, and Z. Li, "Secfednids: Robust defense for poisoning attack against federated learning-based network intrusion detection system," *Future Generation Computer Systems*, vol. 134, pp. 154–169, 2022.

[88] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *International Conference on Machine Learning*, pp. 5650–5659, PMLR, 2018.

[89] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," *Advances in neural information processing systems*, vol. 30, 2017.

[90] S. Awan, B. Luo, and F. Li, "Contra: Defending against poisoning attacks in federated learning," in *Computer Security–ESORICS 2021: 26th European Symposium on Research in Computer Security, Darmstadt, Germany, October 4–8, 2021, Proceedings, Part I 26*, pp. 455–475, Springer, 2021.

[91] N. M. Jebreel and J. Domingo-Ferrer, "Fl-defender: Combating targeted attacks in federated learning," *Knowledge-Based Systems*, vol. 260, p. 110178, 2023.

[92] X. Cao, M. Fang, J. Liu, and N. Z. Gong, "Fltrust: Byzantine-robust federated learning via trust bootstrapping," *arXiv preprint arXiv:2012.13995*, 2020.

[93] B. Biggio, B. Nelson, and P. Laskov, "Poisoning attacks against support vector machines," *arXiv preprint arXiv:1206.6389*, 2012.

[94] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *International Conference on Artificial Intelligence and Statistics*, pp. 2938–2948, PMLR, 2020.

[95] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine learning and systems*, vol. 2, pp. 429–450, 2020.

[96] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," *Advances in neural information processing systems*, vol. 33, pp. 7611–7623, 2020.

[97] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *International conference on machine learning*, pp. 5132–5143, PMLR, 2020.

[98] Q. Li, B. He, and D. Song, "Model-contrastive federated learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10713–10722, 2021.

[99] A. Fallah, A. Mokhtari, and A. Ozdaglar, "Personalized federated learning with theoretical guarantees: A model-agnostic meta-learning approach," *Advances in Neural Information Processing Systems*, vol. 33, pp. 3557–3568, 2020.

[100] S. I. Popoola, R. Ande, B. Adebisi, G. Gui, M. Hammoudeh, and O. Jogunola, "Federated deep learning for zero-day botnet attack detection in iot-edge devices," *IEEE Internet of Things Journal*, vol. 9, no. 5, pp. 3930–3944, 2021.

[101] Y. Fan, Y. Li, M. Zhan, H. Cui, and Y. Zhang, "Iotdefender: A federated transfer learning intrusion detection framework for 5g iot," in *2020 IEEE 14th international conference on big data science and engineering (BigDataSE)*, pp. 88–95, IEEE, 2020.

[102] V. Mothukuri, P. Khare, R. M. Parizi, S. Pouriyeh, A. Dehghantanha, and G. Srivastava, "Federated-learning-based anomaly detection for iot security attacks," *IEEE Internet of Things Journal*, vol. 9, no. 4, pp. 2545–2554, 2021.

[103] D. C. Attota, V. Mothukuri, R. M. Parizi, and S. Pouriyeh, "An ensemble multi-view federated learning intrusion detection for iot," *IEEE Access*, vol. 9, pp. 117734–117745, 2021.

[104] H. Zhu, Y. Zhou, H. Qian, Y. Shi, X. Chen, and Y. Yang, "Online client selection for asynchronous federated learning with fairness consideration," *IEEE Transactions on Wireless Communications*, vol. 22, no. 4, pp. 2493–2506, 2022.

[105] M. Yurochkin, M. Agarwal, S. Ghosh, K. Greenewald, N. Hoang, and Y. Khazaeni, "Bayesian nonparametric federated learning of neural networks," in *International conference on machine learning*, pp. 7252–7261, PMLR, 2019.

[106] Z. Zhu, J. Hong, and J. Zhou, "Data-free knowledge distillation for heterogeneous federated learning," in *International conference on machine learning*, pp. 12878–12889, PMLR, 2021.

[107] A. Z. Tan, H. Yu, L. Cui, and Q. Yang, "Towards personalized federated learning," *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[108] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie, "Class-balanced loss based on effective number of samples," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9268–9277, 2019.

[109] A. K. Menon, S. Jayasumana, A. S. Rawat, H. Jain, A. Veit, and S. Kumar, "Long-tail learning via logit adjustment," *arXiv preprint arXiv:2007.07314*, 2020.

[110] J. Ren, C. Yu, X. Ma, H. Zhao, S. Yi, *et al.*, "Balanced meta-softmax for long-tailed visual recognition," *Advances in neural information processing systems*, vol. 33, pp. 4175–4186, 2020.

[111] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[112] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on knowledge and data engineering*, vol. 21, no. 9, pp. 1263–1284, 2009.

[113] X. Mu, Y. Shen, K. Cheng, X. Geng, J. Fu, T. Zhang, and Z. Zhang, "Fedproc: Prototypical contrastive federated learning on non-iid data," *Future Generation Computer Systems*, 2023.

[114] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, "Federated learning with personalization layers," *arXiv preprint arXiv:1912.00818*, 2019.

[115] P. P. Liang, T. Liu, L. Ziyin, N. B. Allen, R. P. Auerbach, D. Brent, R. Salakhutdinov, and L.-P. Morency, "Think locally, act globally: Federated learning with local and global representations," *arXiv preprint arXiv:2001.01523*, 2020.

[116] L. Collins, H. Hassani, A. Mokhtari, and S. Shakkottai, "Exploiting shared representations for personalized federated learning," in *International Conference on Machine Learning*, pp. 2089–2099, PMLR, 2021.

[117] H.-Y. Chen and W.-L. Chao, "On bridging generic and personalized federated learning for image classification," *arXiv preprint arXiv:2107.00778*, 2021.

[118] H. Lee, S. Shin, and H. Kim, "Abc: Auxiliary balanced classifier for class-imbalanced semi-supervised learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 7082–7094, 2021.

List of Publications

Journals and Transcations

1. T.T. Thein, Y. Ezawa, S. Nakagawa, K. Furumoto, Y. Shiraishi, M. Mohri, Y. Takano, and M. Morii, "Paragraph-based Estimation of Cyber Kill Chain Phase from Threat Intelligence Reports," Journal of Information Processing, Vol. 28, pp. 1025–1029, 2020.

2. T.T. Thein, Y. Shiraishi, and M. Morii, "Malicious domain detection based on decision tree," IEICE Transactions on Information Systems, Vol. E106-D, No. 9, pp.1490–1494, 2023.

3. T.T. Thein, Y. Shiraishi, and M. Morii, "Few-shot Learning-based Malicious IoT Traffic Detection with Prototypical Graph Neural Networks," IEICE Transactions on Information Systems, Vol. E106-D, No. 9, pp. 1480–1489, 2023.

4. T.T. Thein, Y. Shiraishi, and M. Morii, "Personalized federated learning-based intrusion detection system: Poisoning attack and defense," Future Generation Computer Systems, DOI:10.1016/j.future.2023.10.005, 2023.

Doctoral Dissertation, Kobe University
“Study on AI-Powered Cybersecurity for Threat Detection and Mitigation”, 88 pages
Submitted on January 16, 2024
When published on the Kobe University institutional repository /Kernel/, the publication date shall appear on the cover of the repository version.