

Department of Information and Communication Engineering
Graduate School of Information Science and Technology
THE UNIVERSITY OF TOKYO

Master Thesis

Machine Learning Enabled Remote LAN Intrusion Detection

(機械学習を用いた遠隔LANの侵入検知システム)

Zhiqing Zhang

張 智晴

48-196456

Supervisor: Associate Professor Hideya Ochiai

January 2021

Abstract

Security monitoring of remote local area network (LAN) is getting more and more important these days as the cyber attacks shifts the target to the hosts in LANs. However, just the monitoring cannot recognize the differences between vulnerability tests and malware attacks, which may cause confusion among network operators. This thesis proposes an architecture of cloud-based LAN-security monitoring system that can differentiate vulnerability tests and malware attacks happens in the remote LANs. Anomaly & misuse based intrusion detection with machine learning methods has been widely used to unveil malicious behaviors especially for unknown attacks, but there are not so many studies for the behavior of LAN-internal communications. Thus, the first problem is the lack of popular datasets especially captured from real-world LAN-internal communications, and feature extraction for those datasets are also not well designed. Labels for real-world dataset are often difficult to obtain, which limits the use of supervised learning in training based intrusion detection. We designed three intrusion detection methods based on proposed remote LAN monitoring system. First, we designed 3-level detection to classify those activities into ARP scan, TCP port scan, application-level connection establishment and intrusion – the latter two activities are mostly caused by malware not by vulnerability testing. We demonstrate with our prototype implementation that our system can differentiate those behaviors: i.e., vulnerability tests and malware attacks. Second, we proposed clustering based anomaly detection on the dataset collected by our monitoring method. After deploying 45 monitoring devices in distributed LANs in 10 countries, we detected abnormal hosts from total 52,463 hosts appearing during Nov.1st, 2019 to May.5th, 2020 by extracting their behavioral features and classifying them into different clusters. Evaluation results first provide details of our monitoring deployment, and then by comparing different clustering algorithms (K-means, DBSCAN, GMM) they prove that the clustering based detection is capable of recalling 96.0% of confirmed malicious hosts who conducted TCP attacks and 90.7% for UDP attacks, which enables us to believe in the abnormality of other detected hosts. Third, we proposed XGBoost based misuse intrusion detection for LAN. Evaluation results demonstrate that our misuse detection performs 97.5% in overall precision and 97.5% in overall recall. Besides, we also discovered that ARP, MDNS and NBNS are the top 3 protocols that influence the intrusion detection most in LAN.

Contents

Abstract	i
Contents	ii
List of Figures	iv
List of Tables	vi
1 Introduction	1
1.1 Introduction	1
1.2 Structure of this thesis	3
2 Related Works	4
2.1 Honeypot usage in IDS	4
2.2 Network scanners	5
2.3 Network monitoring device	6
2.4 Machine learning based IDS	7
2.4.1 Challenges & points of attention	7
2.4.2 Network traffic datasets	9
2.4.3 IDS systems	9
3 Approaches	12
3.1 Overview	12
3.2 Remote LAN Monitoring System	12
3.2.1 Target on LAN	12
3.2.2 Architecture of monitoring system	13
3.2.3 Design of monitoring node	13
3.2.4 Real data collection	14
3.3 ARP visualization	14
3.4 3-level malware detection	17
3.5 LAN host dataset & feature extraction	19
3.5.1 Necessity of machine learning	19
3.5.2 LAN protocols	19
3.5.3 Host analysis	20
3.5.4 Characteristics of LAN traffic dataset	22
3.5.5 Feature extraction	22

3.6	Clustering based anomaly intrusion detection	23
3.6.1	Potentially malicious hosts	24
3.6.2	Overview	24
3.6.3	Pre-processing	25
3.6.4	Cluster training	26
3.6.5	Clusters division	28
3.7	XGBoost based misuse intrusion detection	28
3.7.1	Labeling	29
3.7.2	Pre-processing	29
3.7.3	Over/under Sampling	29
3.7.4	Model training	30
4	Implementation	32
4.1	LAN monitoring system implementation	32
4.2	Dataset creation	32
5	Experiment & Evaluation	34
5.1	Demonstration of 3-level detection	34
5.1.1	Experiment settings	34
5.1.2	Case 1 and 2: Vulnerability tests	34
5.1.3	Case 3 and 4: Malware attacks	35
5.2	Dataset host analysis	36
5.2.1	Observed hosts	36
5.2.2	Unicast attacks	36
5.3	Evaluation of clustering based anomaly detection	38
5.3.1	Evaluation metrics of clustering-based detection	38
5.3.2	DBSCAN evaluation	38
5.3.3	K-means & GMM evaluation	40
5.3.4	False alarm rate controlling	41
5.3.5	Detection result	43
5.4	Evaluation of XGBoost based misuse detection	44
5.4.1	Experiment settings	44
5.4.2	Classification result	45
5.4.3	Effectiveness of over/under sampling	46
5.4.4	Comparison with other models	46
5.4.5	Feature importance	46
6	Conclusion	47
6.1	Conclusion	47
6.2	Discussion & future Works	47
	List of Publications	49
	Bibliography	50
	Acknowledgments	56

List of Figures

2.1	Architecture of proposed system by Tripathi et al. (Reference: [41])	6
2.2	The two-stage process of self-taught learning: a) Unsupervised Feature Learning (UFL) on unlabeled data. b) Classification on labeled data. (Reference: [14])	11
3.1	Architecture of monitoring system: Nodes are deployed in each LAN for monitoring traffic through it, and sending data file to server for processing and reporting	13
3.2	Malicious host sends SMB packets to monitoring host installed with honeypot .	14
3.3	Design of monitoring node: Honeypots are installed on node covering large range of protocols to differentiate vulnerability testers and malware, and node sends data files to server by ssh	15
3.4	Distributed data collection for Remote LAN Monitoring System	15
3.5	Visualization of ARP requests between hosts in one LAN	16
3.6	Malicious host broadcasts abnormal NBNS queries	21
3.7	Malicious host broadcasts abnormal LLMNR queries	21
3.8	Processing flow for clustering based detection	25
3.9	Binary classification of generated clusters	28
3.10	Dataflow from LAN-traffic database to XGBoost model for misuse detection . .	28
4.1	LAN Monitoring Node: Implemented on Raspberry Pi	33
4.2	Number of monitoring devices deployed in different countries. Background: Average monthly malware encounter rate 2018, Microsoft security intelligence report [66]	33
5.1	Distribution of LANs for each range of number of hosts. LANs where number of hosts observed is greater or equal to a and less than or equal to b is classified in $a - b$ sector.	37
5.2	For each TCP or UDP port, we calculate number of hosts attacking on this port on monitoring device. We only show TCP ports with attacking hosts more than 24 and UDP ports with attacking hosts more than 3.	38
5.3	DBSCAN: Number of clusters under <i>min_points</i> as 100 and different <i>eps</i> . . .	39
5.4	Malicious recall for mixed attack types collected in honeypot-enabled devices on GMM and K-means	40
5.5	Malicious recall for different attack types collected in honeypot-enabled devices on GMM	41
5.6	Malicious recall for different attack types collected in different type of devices on GMM	42

5.7	Malicious recall with different $R\%$ collected in honeypot-enabled devices on GMM	42
5.8	Safe, potentially malicious and confirmed malicious hosts in each monitoring device	43
5.9	Confusion matrix of classification result with XGBoost	43
5.10	Model performance with different sets of sampling strategies	44
5.11	ROC curve on different algorithms	45
5.12	Sum of Feature Importance Score for each Protocol	45

List of Tables

3.1	Dimensions for Host Profiling	20
3.2	Feature Set	23
5.1	Number of confirmed malicious hosts	37
5.2	Number of hosts in each cluster for <i>eps</i> as 0.00536 in DBSCAN	39
5.3	Classification result on different algorithms	44

Chapter 1

Introduction

1.1 Introduction

Local Area Networks(LANs) are functional for interconnected hosts within a limited area (campus, office building, residence, etc.) to share peripheral devices such as printers and communicate with each other by chat or email [1]. In campus, under the control of the universities of institutes LANs provide services for students, faculties and researchers in classroom buildings, laboratories and dormitories [2]. With the limitation on number of hosts and geographical distance, LANs transfer data at a speed much higher than that over a telephone line.

However, cyber attacks towards local area network (LAN) have shown an increasing trend in recent years. By adapting methods like phishing email, malware intrudes in one of innocent hosts in LAN and infects it, posing a serious threat on all hosts with its capability of propagation. Devices may stop working simultaneously and cause a huge impact. Malware intrusion in LAN needs to be detected as early as possible before they make serious attacks after the proliferation.

Network monitoring is a basic approach for detecting such intrusions. However, from the analysis of the network traffic monitoring itself, the system also detects vulnerability test [3] as "intrusion". In network operators' point of view, it is important to detect vulnerability tests and malware attacks separately in order to warn the network administrator with higher priority in case of malware attacks.

This thesis first proposes an architecture of cloud-based LAN-security monitoring system that can differentiate vulnerability tests and malware attacks happened in the remote local area networks. In the architecture, the monitoring node runs several honeypots that are supposed to be interact with malware in the LAN along with basic packet capturing. The server collects the logs of honeypots and packet capture, and our algorithm running on the server recognizes the events.

Further, network Intrusion Detection Systems(NIDS) are usually categorized into two classes. Misuse detection records history attack patterns in a database, with which streaming patterns are matched to determine if an attack appears [4,5]. It is effective and efficient in known attack discovering with high accuracy and low false alarm rate, despite that unknown attacks limit its performance because no related signatures are stored in advance. Anomaly-based detection, in contrast, studies the traffic related to hosts and differentiate abnormal hosts from innocent hosts based on their behavioural patterns [6]. Although it may result in a relatively high possibility of false alarms, proved success in new attack detection enables various implementations for it.

Accuracy of intrusion detection is much enhanced with the development of Machine Learning(ML) methods. These approaches discover deep knowledge and patterns of traffic data and make a distinction between abnormal behaviors and normal ones [7]. Previous works attain high accuracy by applying a wide range of modified learning models on public or self-collected datasets [8–15]. Ensemble models in machine learning techniques meet the requirement of accuracy in intrusion detection with excellent prediction ability and strong adaptability. By combining multiple classifiers, ensemble models overcome the risk of wrong classification result of a single model and promote the overall performance. XGBoost [16] is a gradient tree boosting model widely used in classification and regression tasks. Boosting method enhances the performance by iterative optimization of loss functions while regularization terms in XGBoost control the probability of overfitting.

Nevertheless, it is highly necessary to notice that we are faced with difficulties while applying ML-based detection. First, public datasets, which are commonly adopted in evaluation by previous works, are often criticized for several reasons. The most popular one, KDD dataset family [17, 18] lacks of examples of continuously changing and evolving attack scenarios. Meanwhile, nearly all public datasets are collected in simulated and artificial environments. Applications of models trained on these datasets are seldom tested in real-world networks. These datasets are also created in one network, which may cause a loss of attack pattern diversity. Second, general feature extraction approaches fit in all kinds of networks, but few of them make insights on LAN and design a specific feature set for it. Protocols mainly designed for LAN (NBNS, LLMNR, ARP, etc.) are not paid with enough attention in the behavioral patterns analysis. Third, although in simulated datasets anomaly and normal cases are both clearly labeled in advance, with real-world data unavailability of labels limits the use of supervised learning for detection. We are aware that some of the hosts are infected because their discovered attacks, but in LAN it is our task to detect those hosts who behaves dis-similarly to innocent hosts and prevent them from starting their attacks.

In this thesis, we propose a clustering based anomaly detection on dataset collected by our monitoring method. We deploy honeypot-enabled monitoring devices in remote networks in campus in different countries and the real-time collected traffic data form our academic LAN dataset. Based on the LAN traffic dataset we design host behavioral feature set specific on LANs. In particular for anomaly detection, we take advantage of unsupervised learning methods by applying and comparing different clustering models with designed pre-processing and post-processing so that innocent hosts and abnormal hosts can be classified into different clusters.

In this thesis, we also propose XGBoost based misuse intrusion detection for LAN. Hosts in dataset are with labels defined by ground truth of attacking monitoring devices. We handle the inherent imbalanced classification problem in intrusion detection task with the combination of over and under sampling techniques. Our XGBoost model is trained by real network data with controlling of overfitting problem.

Our monitoring device simulates a benign host with high level of vulnerability and is connected to a switch hub or router in each campus network. Monitored traffic is sent to our central server on a daily basis. We also install honeypots on monitoring devices to enhance interactions with infected hosts. In our dataset we use the data collected from monitoring device in 45 networks in 10 countries within the scope of this paper. We make our intrusion detection on data collected from Nov.1st, 2019 to May.5th, 2020, 187 days in total.

In evaluation of clustering based anomaly detection, We analyze and evaluate three cluster-

ing models: K-means, DBSCAN and GMM on our dataset, and determined GMM at last for its remarkable advantages especially on our dataset. As a result, our detection approach recalls 96.0% of confirmed malicious hosts who conducted TCP attacks and 90.7% for UDP attacks. We also analyze the detected result in each particular network.

Our misuse detection performs 97.5% in overall precision and 97.5% in overall recall. For malicious cases that we value more in result, our approach reaches 93.9% in precision and 97.5% in recall rate. We also compare the performance with 5 machine learning classifiers. Besides, we provide an aggregated importance score result on each protocol and discovered that ARP, MDNS and NBNS are the top 3 protocols that influence the intrusion detection in LAN.

1.2 Structure of this thesis

We first review related works in the field of honeypots, network scanners and intrusion detection systems in Chapter 2. Then we propose our LAN monitoring system and intrusion detection methods: 3-level detection, clustering based anomaly detection and XGBoost based misuse detection in Chapter 3. After that we describe our implementation in Chapter 4 and present experiment results in Chapter 5. We conclude and discuss future works in Chapter 6.

Chapter 2

Related Works

2.1 Honeypot usage in IDS

By representing a real computer system, honeypots are used as a trap for unauthorized communication in network. Classified by level of interaction and deploying environment, honeypots can be categorized into following types: low interaction server honeypots, high interaction server honeypots, low interaction client honeypots, high interaction client honeypots [19]. When used with IDS, honeypots are usually deployed as a server to collect connections from malicious clients. Here we discuss features of several state-of-the-art server honeypots that can be used for intrusion detection.

- Low Interaction Server Honeypots
 - **Dionaea** [20] captures malware by emulating vulnerabilities that malware targets on. It is effective in handling shellcodes and file downloaded from malware. As a Nepenthes successor, it supports TCP port 445 (SMB) protocol, which is frequently attacked in local area network.
 - **Honeytrap** [21] works as a dynamic server honeypot that monitors network packets and handle them by multiple listeners. It is effective in capturing unknown attacks on unknown protocols.
 - **Cowrie** [22] creates a fake file system to log brute force attacks and the entire shell interaction conducted by the attacker. SSH and Telnet protocols supported by Cowrie enable its effectiveness in IoT environment malware and botnet attraction.
 - **IoTPOT** [23] is specifically designed for IoT environments by analyzing Telnet-based attacks against various IoT devices.
- High Interaction Server Honeypots
 - **Argos** [24] was released to automatically identify zero-day attacks. It provides a dynamic taint analysis tool to detect arbitrary control flow attacks and execution attacks.

Recently there are numerous scenarios that honeypots perform as a part of IDS: they can be used to detect malicious activities and profile attackers, and provide IDS with comprehensive

and detailed information of attacks [25]. With their feature of reaction, honeypots extract attack data without mingled with production activity data [19], which could be used in intrusion detection of system. Besides, honeypots can be, and are recommended to be used to combining with other tools in IDSs [25].

Shukla et al. [26] gathered web URLs by deploying high interaction honeypots as clients. After visiting the collected URLs, the malicious ones and labeled and recorded in blacklist for further misuse detection. Chawda et al. [27] reduced the load of high cost by deploying low interaction and network based honeypots as frontend content filters. This design integrate passively collected data and active probes to track attacks. Yashwant et al. [28] proposed honeypot based intrusion prevention system(Buckler) that is capable of promoting overall efficiency in detection of a number of attacks. Agrawal et al. [29] designed honeypot IDS for wireless network malware detection. After passing filters and IDS, network traffic is rerouted to honeypot for deeper investigation. Baykara et al. [30] designed a hybrid honeypot system combining low and high interaction honeypots with IDS for attack detection in real-time network traffic. As it is adaptable with zero-day attacks, the system is effective in both signature and misuse detection.

The main advantage of using honeypot in LAN is that they can profile attacks from both outside and inside. When malware usually propagates in LAN by infecting other hosts, LAN intrusion detection are supposed to be also alert on hijacked hosts. In this scenario, low-interaction type should be used to avoid security risk caused in LAN by real service opening.

Li et al. [31] set up both virtual honeypot and physical honeypot in their system, where the former one works for server protection and the latter for trapping attackers. Lionel et al. [32] noticed the weakness of IoT devices in LAN. Despite of telnet, which is the main target of malware, they found a list of ports that are easy to be attacked on an IoT device. These works concentrate only on honeypot log without any other security tools and methods. Miroslaw et al. [33] monitored both network traffic and attacks logged in honeypots in their system. And from network traffic they found ARP scan from source of gateway and other hosts. But they fail to see the connection between the results of these two tools.

2.2 Network scanners

Various approaches have been put in use for monitoring and analyzing malware intrusion and testing vulnerabilities in existing works. One promising approach is to passively analyze the records of communications from other hosts. Passive analysis provide further protection of malware based on its properties, but deeper investigation can be reached with active connection to source hosts.

Censys [34] utilizes ZMap and ZGrab as pluggable scanners to support full-text queries on remote networks for network researchers. It works as a public query engine and data processing tool with data collected from ongoing Internet-wide scans. Bano et al. [35] designed probes for different layers to scan the liveness and activeness of remote hosts even with firewalls or filtering mechanisms. Antonakakis et al. [36] restricted analysis to remote host banners that were collected within a small window to mitigate the risk of wrongly associating the banner data of uninfected devices with Mirai infections. These works concentrate on the characteristic of the malicious source, but few of these works pay much attention on the vulnerability of specific ports and corresponding service on the targets.

For active port-based analysis, the basic means is port scanning on hosts. Nmap [3], working

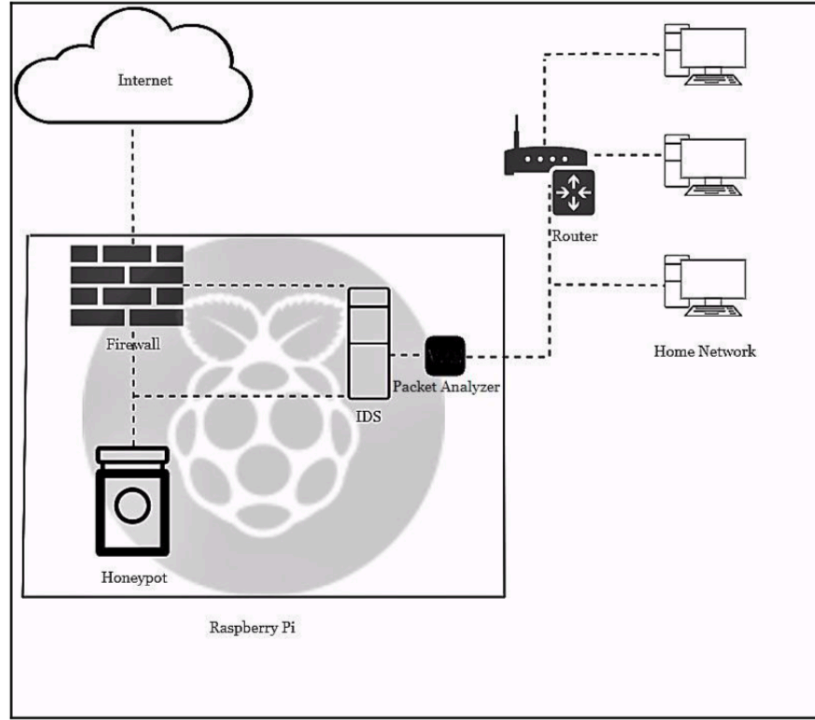


Figure 2.1: Architecture of proposed system by Tripathi et al. (Reference: [41])

as a network scanner, provides network probing service by sending crafted packets to targets and analyzing the response. Some works optimize Nmap for a higher scanning speed. ZMap [37] sends probes as quickly as NIC support with no re-transmission mechanism. However, high speed of probing will cause result influencing packet loss with a saturation on network.

Approaches targeted at specific attack are also attracting attention. For an emerging malware targeted on IoT devices, Mirai Botnet [36], researchers combine multiple powerful and well-established methods including network telescope, active scanning and honeypot for combating for IoT security. By long-term experiment, researchers fully analyze how Mirai Botnet behaves and what degree of damage it can achieve. For UDP cases, traffic measurement and packet analyzing are put in use in DRDoS attacks [38, 39]. As UDP response varies with service, packet analysis considers different protocol for different services [40].

However, these active probes working as vulnerability testers may be detected as malware by intrusion detection systems. From the perspective of scanned network, the unknown scanners behave similarly with malware attempting to attain knowledge of a wide range of networks and waiting for opportunities to intrude. Therefore it is important to differentiate malware from these vulnerability testers for a robust intrusion detection system.

2.3 Network monitoring device

It is straightforward to come up with the idea to connect a laptop or a normal computer inside a single LAN to collect packets by sensing network traffic stream. However, using a laptop host limits the scalability of distributed network traffic collection.

Raspberry Pi is a credit-card sized computer designed to promote computer science education in schools. Due to its small size, it is portable and can be easily carried among cities or even countries without extra efforts. Besides, its low cost and low power consumption enables large scale deployment. This kind of device is ideal for large scale network monitoring.

Tripathi et al. [41] proposed an intrusion detection system based on Raspberry Pi. The architecture of the system is shown in Fig.2.1. They installed intrusion detection system, packet analyzer and honeypot server on the Raspberry Pi device and connected it to the Wi-Fi router with a LAN cable. For intrusion detection, they were using Snort that providing real time packet analysis and signature based intrusion detection service. For Honeypot, they were using Cowrie for SSH and brute force attacker interaction. For packet analyzer, they were using Tshark as a real time network traffic analysis tool.

This is indeed a creative work, yet there are still a number of points need to be improved. From our perspective of view, the main drawbacks of this work include:

- Computational power of Raspberry Pi is incomparable to normal computers. Low level detection methods can be straightly performed on it, but complicated tasks with heavy computational load should be done outside the device.
- Their device contains only third-party tools like Snort and Tshark, lack of customization for specific malware detection.
- They only installed Cowrie honeypot on the monitoring device, leaving ports other than TCP 22(SSH) not listened.
- They have not deployed large scale distributed network monitoring system by this monitoring device.

In our work, we handle the issues existing in this work as well.

2.4 Machine learning based IDS

Success of usage of machine learning in abnormality detection has frequently seen in a wide range of fields. Nowadays researchers immerse themselves in promotion of accuracy as well as expansion of application scenarios based on numerous machine learning approaches. Network intrusion detection systems(NIDS) are long been targeted in use of machine learning owing to the large-amounted data available without careful efforts in data production and prevalent use when situation of cyber-attacks becomes increasingly severe [42–44].

2.4.1 Challenges & points of attention

Although we acknowledge the effectiveness of machine learning based detection method in NIDS, we must clarify that people are often trapped in inappropriate learning, in a scene where learning methods could not match the problem. Specifically, we have to realize the difference of using learning method in NIDS and other scenarios due to the unique features of NIDS [45]. Based in a clarification of potential challenges that researchers may be faced with, we have got a list of advice from previous works. In the following I will first list the possible challenges

and investigate what kind of principles should be remembered before the crazy attempts of algorithms and corresponding parameters.

Lack of data in class-In the problem declaration we have stated that our target for detection concentrates on known attacks in networks, where the function of machine learning is in finding similarities of current occurring network data and previous recorded data [46–48]. With this consideration, often the problem could be converted to a classification problem, "normal" and "abnormal" in 2 classes, or "normal", "abnormal case 1", "abnormal case 2", ..., "abnormal case N" in $N+1$ classes. Quite a few approaches existing require balance of data amount in each amount, as in other scenarios(image classification, natural language word context classification, etc.) spontaneously generate balanced data in each class. State-of-art pre-processing approaches also deal with biased data for better training results. However, in network intrusion detection there is an innate unbalance in class because although cyber attacks are increasingly happening, normal case occupies most time in a network or the normal functioning of network could not be maintained.

Intolerance of False Detection-Accuracy is widely pursued in each scenario of machine learning, however, different scenes tolerate different degree of false detection. Specifically in NIDS, false detection either exerts a huge pressure on network administrators or has a potential to cause severe destruction of the whole network. As false negatives result in the actual intrusion of malware without notification to the administrators, researchers usually choose to sacrifice the rate of false positives for compensation. However, false positives require a large amount of waste time spent by the administrator to figure out who the potential attacker are(who is actually innocent). Even a very small rate of false positives can quickly render an NIDS unusable [49].

Network Diversity-In different networks there is always a certain of difference, which is large enough to influence the actual use of machine learning. Bandwidth, duration of connections and application mix show a large degree of variety among networks. [45, 50, 51] This usually happens when the networks are used in different purposes when NIDS is always supposed to be invented generally suitable for a wide range of networks.

Difficulties in Testing-One of the main challenge of evaluation is the limitation on data of real attacks. Experiments of simulation is effective in basic tests of models, yet intrusion system without real attacks are lack of persuasiveness. Especially in LAN, lack of attack behavior data limits the refining of training model. Publicly available datasets are useful for evaluation despite of their well-known weaknesses, which include 1) time limit when most of them were collected decades ago, 2) environment limit when they are collected in specific networks.

Based on the challenges discussed above there are suggestions proposed by the researchers on how to consider applying machine learning on NIDS.

Keep the scope narrow-On research scope of network intrusion detection system we should fully consider the type of networks. For example, LAN network and public network should not be the clients simultaneously for the same detection system, as the attacks differs severely in behavior pattern. For a host in public network, it receives direct TCP or UDP malicious packets from the global Internet targeted in its IP address, while in LAN malware intrudes into a host without mac address information of other hosts, it first makes a scan in LAN searching for existing hosts and then record mac addresses in its cache and perform further attacks afterwards. The scanning is a pattern in a LAN attack. With this consideration, we should target only on the LAN as our detection scope, which will promote the accuracy and efficiency of detection.

Reducing False Detection-Based on their negative effect, cases of false positives must be a

top priority for an anomaly detection system. As is discussed in previous works, the most step important step towards fewer mistakes is reducing the system's scope. However, for a narrow scope of network there is also a possibility of high level of false positives. To deal with this issue, influencing feature should be selected(or trained), appropriate model should be used, and pre-processing, parameters tuning and post-processing also values in the evaluation.

Evaluation-As is discusses in challenges, one step in necessity is to evaluate trained model(or input training data to) real network traffic. This network should be suffering from real attacks that is within the scope of detection. And the time period of data collection should vary in a relative long period due to the rapid developing and changing of malware. Better method is to establish a database formed by the real attacks, which will assist in 1) further signature-based detection and 2) model adjustment for better accuracy.

2.4.2 Network traffic datasets

Commonly used dataset for intrusion detection can be classified in two big classes, host-based systems where information from defended host machines related to usage of CPU, process, thread and memory etc., and network-based systems where network traffic is being recorded [12]. Network-based datasets are more effective in malware behaviors logging and analysis.

One of the most popular datasets for network-based intrusion detection is KDD99Cup [17], which was originally used for The Third International Knowledge Discovery and Data Mining Tools Competition and collected in a military network environment. Traffic was processed into about 5 million connection records, which are classified into 5 main categories: Safe, DOS, R2L, U2R, probing. It is commonly criticized for large number of duplicated data and out-of-date attack types. Improved version NSL-KDD [18] drops large amount of redundant records and is more frequently used in intrusion detection tasks. Nevertheless, the KDD dataset family is limited in intrusion detection because (1) it is collected in simulated but not real environments, (2) it is collected in only one network when connection variety is lost, (3) it contains only connection records instead of all traffic records and (4) it is too out-dated for modern attack detection. Another widely-used dataset UNSW-NB15 [52] is created at the Australian Center for Cyber Security in 2015 with IXIA traffic generator which simulates 9 families of attacks. It records the whole network traffic with pcap file generated by tcpdump. Comprehensive feature extraction is spotlighted in this work: features include basic, flow-based, content-based, time-based features and labeled features for detection systems.

Network-based datasets, especially those with clear labels of attack types, are mainly based on simulated systems. Although they validate and promote the accuracy of intrusion detection algorithms, they are not capable of reflecting the attacks happening in real-world scenes. Intrusion detection algorithms whose accuracy are proved to be high enough on these datasets are seldom tested in real network intrusion prevention. And as they usually contain traffic collected in the same group of networks, variety of traffic in different networks(and in different locations) is excluded in their considerations.

2.4.3 IDS systems

In prior works, proposed systems demonstrate their ability in intrusion detection. Training-based methods are proved to be working well based on large-scale of data and clear labeled

ground truth.

a. Ensemble learning based IDS

Among systems designed for intrusion detection ensemble learning models obtain better performance using multiple learning algorithms. Since one of the earliest researches in random forest application on misuse detection is proposed by Jiong Zhang, et al. [53], numerous works develop the accuracy, efficiency and universality of intrusion detection systems with random forest and its bagging strategy. Jiaqi Li, et al. [9] proposed two-stage intrusion detection methods: BA (Bat Algorithm) to select typical features and random forest for supervised learning. This work enhances BA to improve its ability for searching optimal features instead of manually extraction. Optimized RF algorithm is applied to classify network traffic into different classes. Different from bagging, boosting based models are exploited in similar tasks for its capability of loss function optimization. Mehrnaz Mazini, et al. [54] proposed intrusion classification with AdaBoost model and artificial bee colony in feature extraction. Arif Yulianto, et al. [55] improved the performance of AdaBoost with PCA in feature reduction and SMOTE for imbalanced problem. After the occurrence of XGBoost [16], Sukhpreet S. Dhaliwal, et al. [56] provided an intrusion detection method with XGBoost on NSL-KDD. Sweta Bhattacharya, et al. [57] further proposed firefly algorithm in feature optimization with XGBoost based intrusion classification.

b. Neural Network based IDS

With the rapidly increasing applications on neural network, NN-based intrusion detection systems also show a success. Chuanlong Yin, et al. [10] applied RNN on NSL-KDD dataset and proved its effectiveness in binary and multiclass classification. Alex Shenfield, et al. [11] experimented ANN on malicious shellcodes detection. Dmitry V. Pantiukhin et al. [12] introduced CNN in detection and improvement for imbalanced data for attack type classification with UNSW-NB15 dataset. Lin Zhang, et al. [13] transform KDD99 network data into an image and applied CNN for image feature extraction and classification.

c. Unsupervised Learning based IDS

Unsupervised learning algorithms are applied in intrusion detection for both feature extraction and classification for unlabeled data. Quamar Niyaz, et al. [14] uses UFL (Unsupervised Feature Learning) to attain a good feature representation from large collection of unlabeled data before applying the learnt feature on labeled data in supervised learning(Fig.2.2). This STL (Self-taught Learning) can be implemented in different combinations of unsupervised and supervised algorithms. This work shows an innovation in the feature selection period of intrusion detection. Artificial selection is relatively weak in performance for the lack of knowledge of feature relevance. Also, unsupervised learning helps in use of unlabeled data, which weights more in the real network. Nour Moustafa, et al. [15] proposed collaborative anomaly detection framework by applying GMM (Gaussian Mixture model) and interquartile range for identifying abnormal patterns.

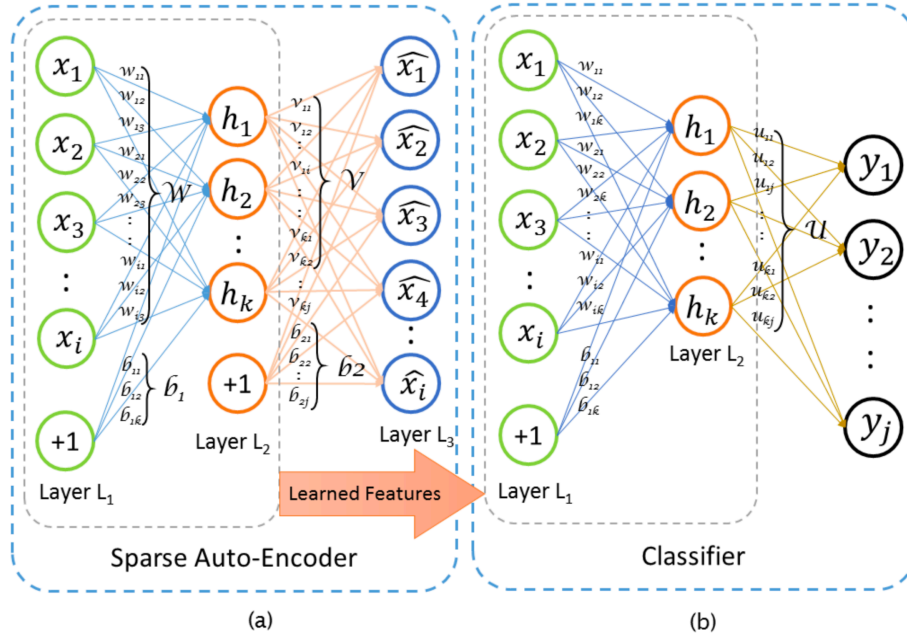


Figure 2.2: The two-stage process of self-taught learning: a) Unsupervised Feature Learning (UFL) on unlabeled data. b) Classification on labeled data. (Reference: [14])

Chapter 3

Approaches

3.1 Overview

This chapter introduces our proposal for remote LAN intrusion detection strategies.

We first provide design of remote LAN monitoring system with monitoring device and processing server. Then we describe our visualization strategy on a specific attack pattern: ARP scan.

With the establishment of LAN monitoring system, we propose three kinds of intrusion detection approaches. 3-level detection detects infected hosts in 3 different level of danger. This method is effective in differentiate malware from vulnerability testers. As we also discovered the effectiveness of machine learning in intrusion detection, we propose two kinds of machine learning based detection methods: clustering based anomaly detection and XGBoost based misuse detection.

3.2 Remote LAN Monitoring System

3.2.1 Target on LAN

Cyber attacks towards local area network (LAN) have shown an increasing trend in recent years. By adapting methods like phishing email, malware intrudes in one of innocent hosts in LAN and infects it, posing a serious threat on all hosts with its capability of propagation. Devices may stop working simultaneously and cause a huge impact. Malware intrusion in LAN needs to be detected as early as possible before they make serious attacks after the proliferation.

Network monitoring is a basic approach for detecting such intrusions. However, from the analysis of the network traffic monitoring itself, the system also detects vulnerability test [3] as "intrusion". In network operators' point of view, it is important to detect vulnerability tests and malware attacks separately in order to warn the network administrator with higher priority in case of malware attacks. This is a typical issue we should consider on discussing on the rate of false positive cases.

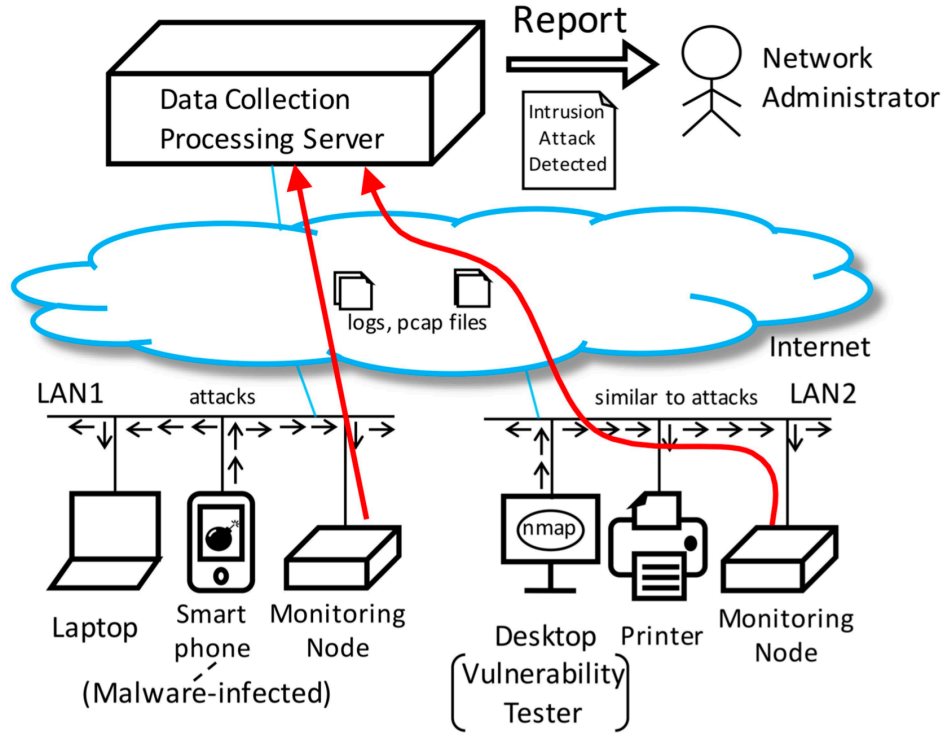


Figure 3.1: Architecture of monitoring system: Nodes are deployed in each LAN for monitoring traffic through it, and sending data file to server for processing and reporting

3.2.2 Architecture of monitoring system

Fig.3.1 shows our design of monitoring system. Our target is to monitor intrusions and malicious activities in each LAN network. We collect data file delivered from monitoring nodes and notify network administrator after processing on server.

Each node is deployed in each LAN network. It is noteworthy that in each LAN, we monitor traffic captured from our monitoring node, but not sniffing packets in the whole network. We connect our node on a normal port but not an advance one(mirroring port), without any requirement and modification of configuration on switch, which enables large-scale deployment of our nodes. Each node is equipped with honeypots for reaction towards attackers. Honeypot logs are sent to the server together with pcap file periodically.

The server collects data files of all nodes, and processes to detect suspicious activities related to ARP scan, TCP port scan, application-level connection establishment and intrusion. The server generates reports and notify their network administrator on abnormality.

3.2.3 Design of monitoring node

We configure tcpdump command to record all traffic related to our node. ARP scan and TCP port scan detection are processed based on pcap file recorded. From ARP requests listed in pcap file, we could understand if ARP scan is happening, and from TCP SYNs listed, we could detect TCP port scan in LAN network.

Honeypot plays a role in verify real intrusion in our system. Normally when a malicious

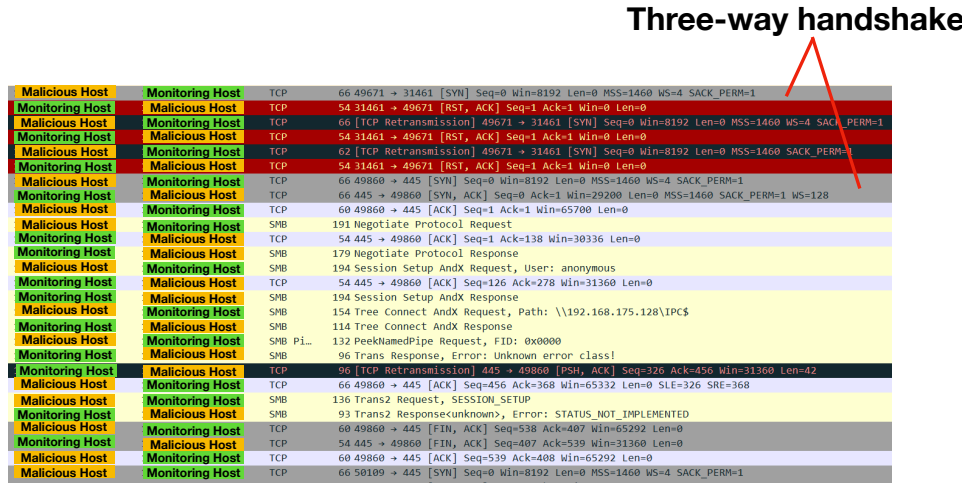


Figure 3.2: Malicious host sends SMB packets to monitoring host installed with honeypot

host detects an existing host by getting its IP address with ARP scan, it directly sends TCP SYN to potential opening ports. If the port is closed and related service is not in use, TCP RST-ACK will be responded. Obviously for our monitoring node, we do not want the connection to fail so easily, in which case we cannot get more information of the intruding host. We want the host to make connections to us after three-handshakes and interact with our node. With this consideration, we install honeypot on the node, as shown in Fig.3.3. Specifically, we use two honeypots reaction. This is because for single honeypot, it covers a limited number of protocols, and we want to monitor more services that might suffer from attacks. Fig.3.2 shows an example that in one network, a malicious host sent TCP SYN packet to monitoring hosts, received the ACK and made the connection because honeypot on TCP port 445 is installed. Then the malicious host sent SMB packets to monitoring host.

For data collection, node connects to the server with SSH, and uses rsync command to deliver pcap file and honeypot logs. As our node are installed in real working LAN environment, it is highly possible that heavy traffic causes large amount of packets saved in pcap and logs. This in turn puts a heavier burden on traffic because of transmission of large files. To solve this problem, we limit transmission rate and utilize idle time (usually at night) for communication.

3.2.4 Real data collection

Up till now the monitoring nodes is distributed to 22 universities or institutes in 11 countries in the method shown in Fig.3.4. On their local networks we have already detected and reported real malware attacks, with the help of that local network administrators have checked the reports and protected their networks. The collaborators provide us with real network environments, from which further researches are much easier to develop.

3.3 ARP visualization

There is a special phenomenon happening in LAN that malicious hosts (and some vulnerability testers) often apply ARP scan to search existing hosts in the network. The original purpose

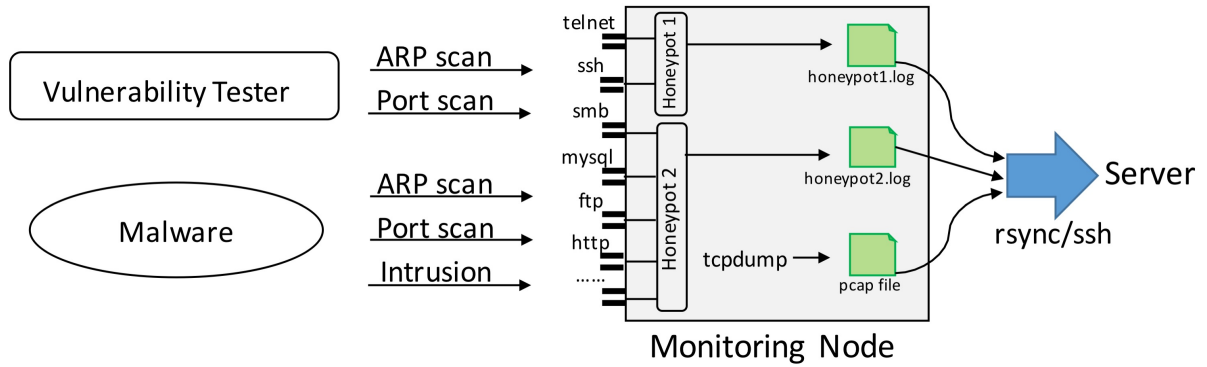


Figure 3.3: Design of monitoring node: Honeypots are installed on node covering large range of protocols to differentiate vulnerability testers and malware, and node sends data files to server by ssh

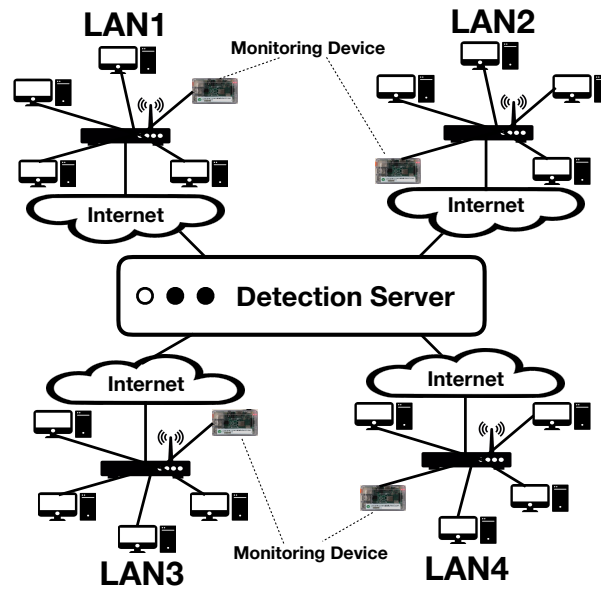


Figure 3.4: Distributed data collection for Remote LAN Monitoring System

of the ARP protocol is to find the MAC address of another host with a targeted IP address. Specifically, when a host A wants to know the MAC address of IP IP_B , then A broadcasts an ARP request packet querying "Who is IP_B ? Tell IP_A ". If host B exists in LAN and receives this broadcast packet, then it will respond its MAC address to host A . Host A then records the MAC address in its ARP cache for further packets sending. This mechanism is also applied by network administrators and vulnerability testers. For example, if the admin wants to know which hosts are connected to LAN, an ARP scan(ARP requests with a range of IP requests) is broadcasted.

However, this mechanism also provides convenience for a network intruder. After the intruder enters the LAN, it knows nothing about the information of other benign hosts. For further malware propagation, it will first make an ARP scan to acquire MAC addresses of existing hosts, storing the information in the cache. When the time is ripe(for example, receiving the attack command from a C&C server), then it sends unicast packets to the benign hosts for malware

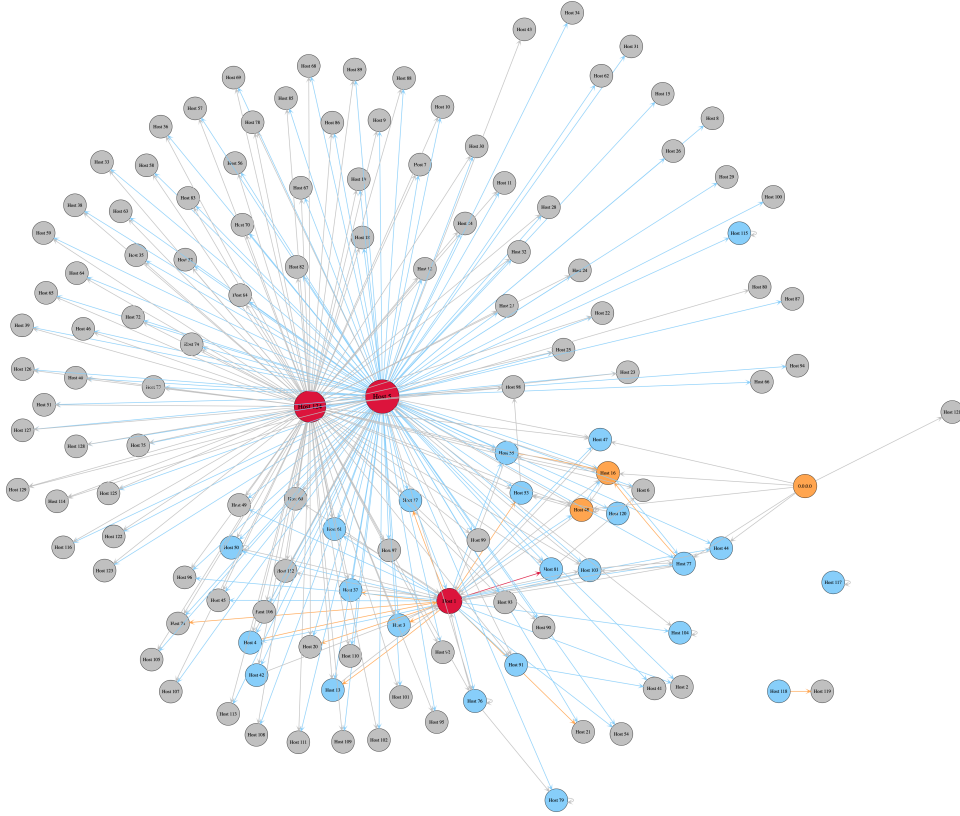


Figure 3.5: Visualization of ARP requests between hosts in one LAN

propagation.

To visualize this phenomenon, we design an ARP visualization strategy. Fig.3.5 is an example.

- **Vertices** stand for hosts in LAN. We define the out-degree of vertices according to the definition in graph theory.
 - Red vertices: $outdegree > 20$
 - Orange vertices: $3 < outdegree \leq 20$
 - Blue vertices: $0 < outdegree \leq 3$
 - Grey vertices: $outdegree = 0$
- Grey vertices are considered as non-existing hosts. If a host sends ARP requests to large amount of non-existing hosts, then it should be considered that the host is making an ARP scan.
- **Edges** stand for ARP request packets. The color of the edge describes number of packets in an ARP request.
 - Red edges: $\#packet > 900$
 - Orange vertices: $100 < \#packet \leq 900$

- Blue vertices: $5 < \#packet \leq 100$
- Grey vertices: $0 < \#packet \leq 5$

3.4 3-level malware detection

This section proposes a 3-level LAN intrusion detection method that can differentiate vulnerability tests and malware attacks happened in the remote local area networks. In order to propagate into other hosts or steal important information from its databases in a LAN, malware tries to find open TCP ports from all the hosts in the network. In this process, it broadcasts ARP requests to lookup the MAC address from all the IP addresses one-by-one, which we call ARP scan, in this thesis. Then, it sends TCP SYNs to potentially available ports, which is TCP port scan. If it gets TCP SYN+ACK from the target host, it means that the TCP port is open. This behaviour is actually similar to nmap – vulnerability tester.

Malware, then, establishes the connection to the target host at the application layer, and tries several username and password pairs, i.e., brute-force attacks to login. And, after getting privileges to work on the platform, it downloads files into the platform or steal data from the database. These actions are almost unique to malware's behavior. We recognize that there are some vulnerability tester that behaves in similar way [58], but they would be deployed with extensive attentions.

We set up a server for collecting data from LANs and processing for reporting to network administrator. Our server has two interfaces: 1) Data collection, for receiving pcap files and honeypot logs from all nodes in their LAN with SSH, and 2) Reporting, for notifying administrator on the activities that we consider malicious by email. Main function of our server is to detect abnormal activity in each LAN, including ARP scan, TCP port scan, application-level connection establishment and intrusion, based on the relationship of these activities.

Specifically, we classify hosts with abnormal activity into 3 types.

1) Host with ARP scan. This kind of host makes sudden ARP scan to other host in the LAN, to get their MAC addresses by broadcasting IP address. But they have not done any TCP related behavior until now. This kind of host can be malicious, but also can be intentional, as vulnerability tests often includes ARP scan after network configuration. Let A be the set of source IP addresses who make ARP scan in the LAN.

2) Host only with TCP port scan and ARP scan. From the list of hosts that make ARP scan, there is a subset that makes TCP SYNs to ports on detected hosts. Let T be the set of source IP addresses that made TCP SYNs to the monitoring node, and T_A be the set of IP addresses that made TCP SYNs after ARP scan.

$$T_A = A \cap T$$

This kind of host can be also innocent, as there is the case when vulnerability testers makes TCP port scan with tools like nmap, after modification on network configuration or even periodically.

3) Host with application-level connection establishment or intrusion after ARP scan. This hosts should be considered malicious, because after ARP scanning hosts in LAN and found ones that exist, the scanning host makes connection into the discovered host with several kinds of behavior. The behaviors include but not limited to brute-force connections, trying user name

and password to login, executing command or script and downloading malicious files on the discovered host. Let H be the set of hosts making connection and trapped into honeypots, we define this kind of host as:

$$H_A = A \wedge H$$

Here we demonstrate our method on detecting different behaviors on each node. Algorithm 1 shows our method of setting up a dictionary for honeypot events. With this structure modification, we can access to sessions created from a source IP in $O(1)$ time complexity.

Algorithm 1 HONEYPOT_LOG_PROCESSING(log)

```

1:  $HP\_dict = \{\}$ 
2: for  $event$  in  $log$  do
3:    $HP\_dict[event.src\_ip][session\_id].add(event)$ 
4: end for
5: return  $HP\_dict$ 

```

Algorithm 2 DATA_FILE_HANDLER($pcap, log$)

```

1:  $A = DETECT\_ARP\_SCAN(pcap)$ 
2:  $REPORT\_ARP\_SCAN(A)$ 
3:  $T_A, T_A\_packets = \{\}$ 
4:  $H_A, H_A\_sessions = \{\}$ 
5: for  $p \in pcap$  do
6:   if  $p.TCP.SYN$  and  $\neg p.TCP.ACK$  and  $p.ip.src\_ip$  in  $A$  then
7:      $T_A.add(p.ip.src\_ip)$ 
8:      $T_A\_packets[p.ip.src\_ip].add(p)$ 
9:   end if
10: end for
11:  $REPORT\_TCP\_PORT\_SCAN(T_A, T_A\_packets)$ 
12:  $HP\_dict = HONEYPOT\_LOG\_PROCESSING(log)$ 
13: for  $IP$  in  $T_A$  do
14:   if  $IP$  in  $HP\_dict$  then
15:      $H_A.add(IP)$ 
16:      $H_A\_sessions[IP].add(HP\_dict[IP])$ 
17:   end if
18: end for
19:  $REPORT\_TCP\_INTRUSION(H_A, H_A\_sessions)$ 

```

Algorithm 2 shows our main process of classification and reporting. A is detected with fitting model of degree of ARP request trend [59]. Degree of ARP request means the number of target hosts to which ARP request is sent by a source host. By storing history degree value in database and applying fitting algorithm, sudden increasing of degree, which means abnormal ARP scan, could be detected. We define p as a packet in $pcap$, and from TCP SYN packets we filter hosts with TCP port scan. From honeypot log dictionary we filter hosts making connections to target host. We report the behaviors separately.

3.5 LAN host dataset & feature extraction

3.5.1 Necessity of machine learning

By now before this section, we are still discussing basic detection methods for intrusion detection. We believe that usage of machine learning will assist in great progress of the target, for the reasons as follows:

a. Narrow Scope of Network

The target we have chosen is LAN network, which is a narrow scope to detection target for the relatively fixed feature of intrusion. Although the types of attacks vary, they may happen in each one of the LANs in the world, relieving the regional diversity. Also, we consider the attacks happening recently, with less consideration of past happening attacks(which is the usual case for public dataset).

b. Enough effective data

As we have stated, our data is collected in real scenes in the real networks of universities and institutes. Most of the data is from student networks suffering from a high level of probability of intrusion happening. According to our current data we have already detected real malware and validated from the administrators. We are still devoting in distributing the nodes to our collaborators for more kinds of attacks for detection.

c. Accuracy Promotion

One unsatisfying result of current detection method is the high level of false positive rate. As we have discussed in this survey, well-trained machine learning models are responsible for lowering false positive rate to reach high accuracy. We believe using machine learning in detection will result in a promotion of detection accuracy.

3.5.2 LAN protocols

Despite of the fact that they all follow the hierarchical network model, LANs are different from global networks in the specific protocols where hosts are adopting for different purposes of communication. Analysis of communications of LANs is based on these protocols. We can classify protocols frequently seen in LAN packets by layers in OSI model they belong to as:

- **Application Layer protocols:** DHCP, SSDP (Simple Service Discovery Protocol), MDNS (Multicast DNS), LLMNR (Link-Local Multicast Name Resolution), BROWSER (Microsoft Windows Browser Protocol)
- **Session Layer protocols:** NBNS (NetBIOS Name Service)
- **Transport Layer protocols:** TCP, UDP
- **Network Layer protocols:** ICMP, IGMP
- **Data Link Layer protocols:** ARP

3.5.3 Host analysis

Table 3.1: Dimensions for Host Profiling

Protocol Type	Protocol	Item	Explanation
Unicast	TCP	(port, SYN packet numbers, packet numbers) list	Number of SYN packets and all TCP packets sent to monitoring node on each port
	UDP	(port, packet numbers) list	Number of all UDP packets sent to monitoring node on each port
UDP Broadcast/ Multicast	DHCP	packet numbers	Number of DHCP packets
	SSDP	packet numbers	Number of SSDP packets
	BROWSER	packet numbers	Number of BROWSER packets
	MDNS	packet numbers	Number of MDNS packets
		query numbers	Number of distinct MDNS queries
	NBNS	packet numbers	Number of NBNS packets
		query numbers	Number of distinct NBNS queries
		abnormal query numbers	Number of distinct NBNS queries that are abnormal
	LLMNR	packet numbers	Number of LLMNR packets
		query numbers	Number of distinct LLMNR queries
		abnormal query numbers	Number of distinct LLMNR queries that are abnormal
	Others	(port, packet numbers) list	Number of packets sent on each port
Network Layer	ICMP	packet numbers	Number of ICMP packets
	IGMP	packet numbers	Number of IGMP packets
Data Link Layer	ARP	in-degree	Number of distinct hosts that receive ARP requests from
		out-degree	Number of distinct hosts that send ARP requests to
		multi-degree	Number of distinct hosts that send ARP requests to and receive from simultaneously

Infected hosts, whose main purpose is to affect the normal use of other devices/the whole network by spreading the malware in LAN, behave abnormally when comparing with other hosts. The behaviors include a range of protocols with broadcast, multicast and unicast services.

Each pcap file records the network traffic captured by monitoring node within one day. Considering hosts to be our detection object, we profile each host in the dimensions described in Table.3.1. These dimensions are designed specially for hosts in LAN, when some of the protocols(ARP, NBNS, etc.) are only available in LAN.

A host tries to search the MAC address of other hosts in LAN by ARP protocol. By making a consecutive ARP request scanning on IP address space, a host records the MAC addresses of the hosts who has responded in its ARP cache. Then TCP or UDP unicast packets, which may be malicious, can be directly sent to those recorded hosts afterwards. Numerical representation

Source	Destination	Protocol	Length	Info
Malicious Host	Broadcast	NBNS	92	Name query NB RJXBTRBVZOMKWFx<00>
Malicious Host	Broadcast	NBNS	92	Name query NB ERYJBNGKHVLZXN<00>
Malicious Host	Broadcast	NBNS	92	Name query NB EWDYQRPXPMUVU<00>
Malicious Host	Broadcast	NBNS	92	Name query NB RJXBTRBVZOMKWFx<00>
Malicious Host	Broadcast	NBNS	92	Name query NB ERYJBNGKHVLZXN<00>
Malicious Host	Broadcast	NBNS	92	Name query NB EWDYQRPXPMUVU<00>
Malicious Host	Broadcast	NBNS	92	Name query NB RJXBTRBVZOMKWFx<00>
Malicious Host	Broadcast	NBNS	92	Name query NB ERYJBNGKHVLZXN<00>
Malicious Host	Broadcast	NBNS	92	Name query NB EWDYQRPXPMUVU<00>
Malicious Host	Broadcast	NBNS	92	Name query NB PYTQBREDPGRPINJ<00>
Malicious Host	Broadcast	NBNS	92	Name query NB HFSNAXV<00>
Malicious Host	Broadcast	NBNS	92	Name query NB MEBLPRWPBEWA<00>
Malicious Host	Broadcast	NBNS	92	Name query NB KHPZNVPCUV<00>
Malicious Host	Broadcast	NBNS	92	Name query NB FMUUAVFCXZ<00>
Malicious Host	Broadcast	NBNS	92	Name query NB OMZRKKXDTSYICWQ<00>
Malicious Host	Broadcast	NBNS	92	Name query NB PYTQBREDPGRPINJ<00>
Malicious Host	Broadcast	NBNS	92	Name query NB HFSNAXV<00>
Malicious Host	Broadcast	NBNS	92	Name query NB MEBLPRWPBEWA<00>
Malicious Host	Broadcast	NBNS	92	Name query NB KHPZNVPCUV<00>
Malicious Host	Broadcast	NBNS	92	Name query NB FMUUAVFCXZ<00>
Malicious Host	Broadcast	NBNS	92	Name query NB OMZRKKXDTSYICWQ<00>
Malicious Host	Broadcast	NBNS	92	Name query NB MEBLPRWPBEWA<00>
Malicious Host	Broadcast	NBNS	92	Name query NB HFSNAXV<00>
Malicious Host	Broadcast	NBNS	92	Name query NB PYTQBREDPGRPINJ<00>
Malicious Host	Broadcast	NBNS	92	Name query NB KHPZNVPCUV<00>
Malicious Host	Broadcast	NBNS	92	Name query NB FMUUAVFCXZ<00>

Figure 3.6: Malicious host broadcasts abnormal NBNS queries

Source	Destination	Protocol	Query Contents
Malicious Host	Broadcast	LLMNR	Standard query 0x06a5 A ngmitubh
Malicious Host	Broadcast	LLMNR	Standard query 0xe5ab A myibucr
Malicious Host	Broadcast	LLMNR	Standard query 0xd5c1 A zcfbhxc
Malicious Host	Broadcast	LLMNR	Standard query 0x06a5 A ngmitubh
Malicious Host	Broadcast	LLMNR	Standard query 0xe5ab A myibucr
Malicious Host	Broadcast	LLMNR	Standard query 0xa66c ANY user-PC
Malicious Host	Broadcast	LLMNR	Standard query 0xa66c ANY user-PC
Malicious Host	Broadcast	LLMNR	Standard query 0xaad5 A isatap
Malicious Host	Broadcast	LLMNR	Standard query 0xaad5 A isatap
Malicious Host	Broadcast	LLMNR	Standard query 0x9748 A qagisucvnx
Malicious Host	Broadcast	LLMNR	Standard query 0xcfe4 A nzbvtzyjehk
Malicious Host	Broadcast	LLMNR	Standard query 0x3ff7 A uaacitycusqapy
Malicious Host	Broadcast	LLMNR	Standard query 0x9748 A qagisucvnx
Malicious Host	Broadcast	LLMNR	Standard query 0xcfe4 A nzbvtzyjehk
Malicious Host	Broadcast	LLMNR	Standard query 0x3ff7 A uaacitycusqapy
Malicious Host	Broadcast	LLMNR	Standard query 0x76fc ANY user-PC
Malicious Host	Broadcast	LLMNR	Standard query 0x76fc ANY user-PC
Malicious Host	Broadcast	LLMNR	Standard query 0x3195 A jkylccgmupifwu
Malicious Host	Broadcast	LLMNR	Standard query 0x136b A kbphieoswioqzc
Malicious Host	Broadcast	LLMNR	Standard query 0x9dbb A cpxlittubym
Malicious Host	Broadcast	LLMNR	Standard query 0x3195 A jkylccgmupifwu
Malicious Host	Broadcast	LLMNR	Standard query 0x136b A kbphieoswioqzc
Malicious Host	Broadcast	LLMNR	Standard query 0x9dbb A cpxlittubym
Malicious Host	Broadcast	LLMNR	Standard query 0x56aa A isatap

Figure 3.7: Malicious host broadcasts abnormal LLMNR queries

of ARP scanning applies the "degree" in graph theory.

For item *abnormal query numbers* in NBNS and LLMNR, we use the ratio of vowels in the query name to decide it is abnormal. For each query, the ratio between the number of vowels and the query length is calculated. It is believed that the ratio of vowels in legitimate queries is higher than that in abnormal queries. For example, *workgroup* and *paraview* (vowels ratio as 0.33 and 0.5) are legitimate queries while *RJXBTRBVZOMKWFx* and *ERYJBNGKHVLZXN* (vowels ratio as both 0.07) are abnormal in Fig.3.6. A threshold of

0.3 is set to decide if the query is abnormal or not. We apply the same strategy for LLMNR protocol (Fig. 3.7).

These dimensions of host in LAN are effective in both misuse detection and anomaly based detection. In misuse detection, the profile of malicious host is recorded after it is confirmed to have done attacks. After that if another host who has a very similar profile with those in the database appears, we believe it belongs to the same behaviour pattern and assert its dubiousness. Anomaly detection is available with these dimensions because malicious hosts usually behave abnormally (although we may not have seen the behavioral pattern before) when compared to normal hosts.

3.5.4 Characteristics of LAN traffic dataset

In Chapter 2 we reviewed the pros and cons of commonly used datasets for intrusion detection systems. With our configuration of data collection in Asia academic LANs and analysis on hosts, we clarify the characteristics of our dataset:

- **From Real World Environments** While most of intrusion detection algorithms benefit from simulated attacks in their performance, we collect data real world networks which are continuously being used for a variety of purposes. Real attacks are aimed to be detected with limited information in ground truth.
- **From Remote & Distributed Environments** We do not limit our data collection in one network, one laboratory or even one country. Different locations suffer from different distributions of attack types. We enhance the robustness of intrusion detection with these distributed data. Also it guarantees universality for algorithms based on this dataset as they could be applied in different countries.
- **Up-to-date** With the continuous evolution of attack patterns, outdated datasets are becoming less powerful in detection because of the emergence of modern attacks. Our dataset include newest traffic containing up-to-date attacks.
- **Real-time Updating** Monitoring devices deployed in networks of our collaborators are persistently collecting data and uploading to our server. We can always update our dataset with the newly collected data. Detection algorithms are also enabled to provide real-time results for network administrators to manage their networks.
- **Suitable for Both Misuse Detection and Anomaly Detection** As is described in Section 3.5.3, our designed profiling dimensions are available for both signature based detection for known attack patterns and anomaly based detection for unknown attack patterns.

3.5.5 Feature extraction

According to our data collection configuration, we monitor hosts in LAN in consecutive days. To develop a feature set that can well describe behaviors of hosts in LANs, we design features in a combination of time dimension and protocol dimension.

In protocol dimension, we extract features related to protocols of packets sent by the hosts based on Table 3.1. In time dimension, we calculate the average and maximum of each protocol

dimension item on time domain. Average values can be effective because they demonstrate how a host normally behaves and maximum values matter because they show a sudden anomaly in a specific day. In our feature set, average values are based on total time the host exists in LAN, with granularity as 6 hours. Our feature set is described in Table 3.2. Explanations for each feature can be referenced in Table 3.1.

Table 3.2: Feature Set

Feature Set
existing days
avg/max ARP in-degree
avg/max ARP out-degree
avg/max ARP multi-degree
avg/max ICMP packet numbers
avg/max IGMP packet numbers
avg/max DHCP packet numbers
avg/max SSDP packet numbers
avg/max BROWSER packet numbers
avg/max LLMNR packet numbers
avg/max LLMNR query numbers
avg/max LLMNR abnormal query numbers
avg/max NBNS packet numbers
avg/max NBNS query numbers
avg/max NBNS abnormal query numbers
avg/max MDNS packet numbers
avg/max MDNS query numbers

We do not use TCP & UDP unicast information in feature set because it is responsible for checking confirmed malicious hosts. The detected confirmed malicious hosts are used in our evaluation.

3.6 Clustering based anomaly intrusion detection

In this section, we propose a clustering-based intrusion detection methods. We first provide our classification of abnormal hosts, based on which we propose our detection flows: feature extraction, pre-processing, cluster training and cluster division. We show how we design each link separately in this section. For cluster training we discuss three prevalent clustering algorithms from different method groups: K-means, DBSCAN and GMM, and analyze difficulties we are faced with on applying these methods on our academic LAN dataset.

3.6.1 Potentially malicious hosts

Based on the dimensions of behavioral profiling of hosts, we are capable of tracking the behaviors of each host in our dataset. With large number of samples and high dimensions of features, learning based anomaly detection methods are widely used as described in Chapter 2. Supervised learning is typical effective when labels of normal cases and abnormal cases are clearly defined. These algorithms are normally used in simulation based dataset (NSL-KDD, UNSW-NB15, etc.) because they have clear definition not only on whether the case is malicious but on attack types as well. They are proved to be able to give a high accuracy (and also well-balanced precision & recall rate) in malicious case detection and attack type classification.

However, labels are not always clear in real world environment. What we are clear about are TRUE cases: we realize one of the hosts in LAN is malicious when our monitoring device suffers from attacks from it. This attack usually appears in a unicast TCP & UDP manner from the malicious host to monitoring device. However, we cannot have assumptions on the innocence of other hosts, because (1) some hosts may be malicious but they are waiting for a timing (receiving commands from C&C server, etc.), or (2) they do not include monitoring device in their attack list. What we declare is that malicious hosts, whether attacking monitoring device or not, are highly likely to behave abnormally during their stay in LAN. This is the foundation of our learning based detection methodology.

With this consideration, we do not assume labels on hosts in data set. We apply one group of unsupervised learning algorithms-clustering in anomaly detection. Based on assumption that most hosts in LAN are safe, we believe that large clusters are formed by benign hosts while hosts in small clusters are considered to be abnormal.

According to our definition, group of hosts detected to be with abnormal behaviors (called abnormal hosts in this paper) can be divided into two subgroups: Confirmed malicious hosts who have been seen attacking monitoring device and potentially malicious hosts.

Confirmed malicious hosts are easily to be detected by their attack activities, which we interpret as sending TCP & UDP unicast packets to monitoring device. Although vulnerability testers may also send unicast packets, network administrators tend to log their information in a white-list, which is beyond the scope of this thesis.

We naturally base validation of our detection method on confirmed malicious hosts detection. If nearly all of confirmed malicious hosts are included in detected abnormal hosts, we believe our detection method is effective in finding potentially malicious hosts.

3.6.2 Overview

Fig.3.8 shows the processing flow for clustering based malicious hosts detection. We first extract features for hosts in database based on dimensions of host profiling described in Section 3.5.3. After pre-processing, where we conduct scaling and PCA on our feature matrix, we train the clustering model to get clusters. We make a division between safe and abnormal clusters after that. As we have described in Section 3.6.1, we determine confirmed malicious hosts and potentially malicious hosts within abnormal hosts.

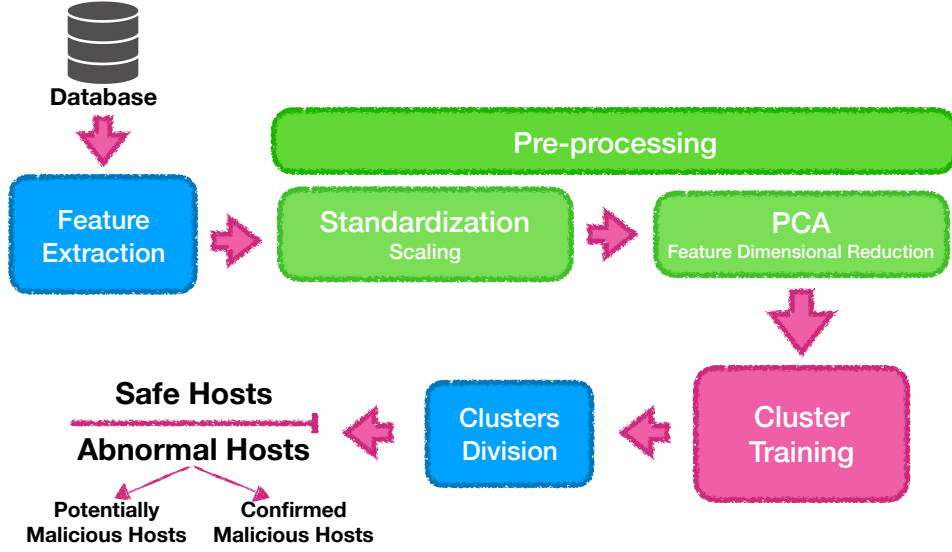


Figure 3.8: Processing flow for clustering based detection

3.6.3 Pre-processing

a. Feature Scaling

As magnitudes and ranges vary in different features, we perform feature scaling to match the scale of each feature column. Especially when we are applying distance based clustering method, difference of scales in different features matter when using Euclidean distance.

The fact that not all hosts send packets in a specific protocol leads to appearance of zeros in feature matrix. Actually, as shown in experiments we found zeros occupy a large proportion in which case we call it a sparse matrix. Not to destroy the sparsity of original feature matrix, we apply *MaxAbsScaler* algorithm here:

$$X_scaled[axis = k][i] = \frac{X[axis = k][i]}{\max |X[axis = k]|}$$

For each feature k independently, we divide each value by the maximum of absolute values of feature k . As our feature matrix contains zero or positive values only, the scaled range becomes $[0,1]$, when zeros are still mapped to zeros.

b. Principle Component Analysis

To reduce the feature dimensions while retaining as much information as possible, we apply the widely used PCA (Principle Component Analysis) on our feature matrix. It combines highly correlated variables together to form a smaller number of feature set that count for most variance in the original data. By assigning ratio p , PCA selects the number of components such that the amount of variance that needs to be explained is greater than ratio p . We set p as 0.99 for our processing.

3.6.4 Cluster training

Commonly used clustering algorithms can be divided into different groups according to their fundamental theory. In this thesis, we discuss theories of three kinds of clustering methods and apply one algorithm in each group on our classification task.

a. K-means: Partition-based Method

The basic concept of partitioning-based clustering method is to discover K groups in the dataset by optimizing a specific object function, where Sum of Squared Errors(SSE) is commonly applied.

When we have K clusters in a partition C and for each cluster C_k we have its centroid c_k . To optimize SSE , it is to minimize the sum of all the squared distance from data point x_i to its centroid.

$$SSE(C) = \sum_{k=1}^K \sum_{x_i \in C_k} ||x_i - c_k||^2$$

A data partition is made in initialization and the partition is improved by iterative optimizing the object function. To find a global optimal instead of exhaustively enumerating all partitions, more realistically a heuristic method(for example, K-means) is applied.

In K-means [60], each cluster is represented by the center of the cluster. First it randomly selects the K points as initial centroid. They form K clusters by assigning each point to its closest centroid, and new centroid of the clusters are re-computed. And repeatedly centroids and clusters are updated until it reaches convergence criterion.

K-means is a pretty efficient algorithm for clustering because the time complexity is linear to the size of the number of data points. Cluster training will be time-saving when we have increasing dataset size with real-time traffic collection.

There are also concerns when applying K-means on our academic LAN dataset. (1) The number of clusters K need to be specified in advance. In practice we often run a range of K values to select the best one. And in Section. we evaluate performance for each K . (2) Because it uses the sum of the squared distance, K-means performs not well in finding clusters with non circular shapes, which may pose a problem when we are not clear about shapes of each cluster of hosts.

b. DBSCAN: Density-based Method

Density-Based Spatial Clustering Applications with Noise (DBSCAN) [61] is the most representative algorithm in density-based clustering. It works by defining a cluster as the maximal set of density-connected points. It has two parameters that are worth taking into account: ϵ , the maximum radius of the neighborhood, and min_points , the minimum number of points in the ϵ -neighborhood to define a cluster. Three kinds of points in DBSCAN include (1) Core points, having at least min_points points within its ϵ -neighborhood, (2) Border points, having less than min_points points within its ϵ -neighborhood but existing in the neighborhood of a core point, (3) Outlier points that cannot be reached by a cluster.

DBSCAN first selects a random non-core point and determine it is a core point. Then it adds all directly density-reachable points to its cluster, and conducts neighbor jumps to each indirectly density-reachable points and add them to the cluster also. Added outlier points are labeled as border points. These steps are repeated until all points are assigned to a cluster or labeled as outlier.

DBSCAN is capable of distinguishing noise from cluster groupings. It can detect outlier hosts and filter them out in clustering. And DBSCAN can model all shapes of clusters, not limiting them as shaped in circular. Also, DBSCAN does not require the pre-defined number of clusters K .

The particular disadvantage of DBSCAN is that it cannot handle the cases when in each clusters the density differs seriously well. Difference in cluster densities require different ϵ value, but in DBSCAN parameter ϵ is pre-defined and fixed. We consider the densities of clusters with abnormal hosts are smaller than those for safe hosts, because safe hosts maintain low values of each feature while abnormal hosts may have an apparently large value in a specific feature, which contributes to large distances between each other.

c. GMM: Model-based Method

In model-based method [62], we assume a clustering consists of K probabilistic cluster C_1, \dots, C_K with probability density functions f_1, \dots, f_K respectively and their corresponding probabilities are w_1, \dots, w_K . The probability of a data point x_i is generated by C_j is

$$P(x_i, C_j) = w_j f_j(x_i)$$

Then the probability of x_i is generated by th set of cluster C is

$$P(x_i|C) = \sum_{j=1}^K w_j f_j(x_i)$$

On assuming the data points in dataset D are generated independently, the probability that all points in dataset is generated by C is

$$P(D|C) = \prod_{i=1}^n P(x_i|C) = \prod_{i=1}^n \sum_{j=1}^K w_j f_j(x_i)$$

The task of model-based clustering is to find K probabilistic clusters(set C) for dataset D that maximize $P(D|C)$. Each cluster is mathematically represented by a parametric distribution. Mixture of Gaussian distributions is commonly used, for which we call Gaussian Mixture Model(GMM) [63].

In GMM, we assume all the clusters are formed based on Gaussian distributions with different parameters (i.e. means and co-variance matrices). Maximum likelihood estimation(MLE) is applied to maximum the $P(D|C)$ with specific set of Gaussian parameters. In practice EM approach is used for finding parameters in MLE.

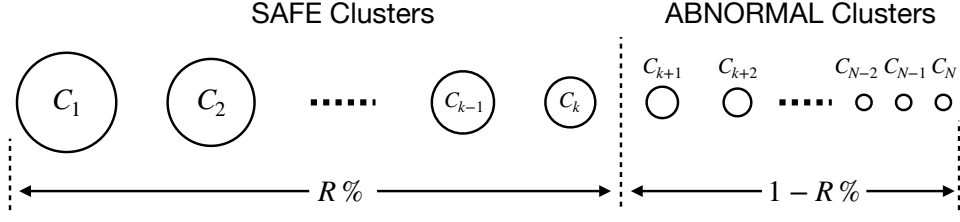


Figure 3.9: Binary classification of generated clusters

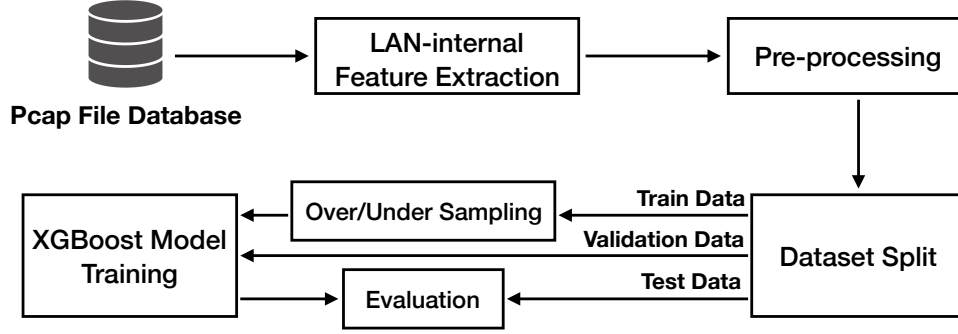


Figure 3.10: Dataflow from LAN-traffic database to XGBoost model for misuse detection

GMM does not assume the shape of clusters in advance, which means it can handle sophisticated and non-convex shaped clusters. Concerns of using GMM in our dataset include overfitting problem when we assign excessive number of clusters as the mixed model will become too complex. We also need to assign number of clusters K in advance.

These three kinds of methods all show advantages and disadvantages on applying to our dataset. In Chapter 5 we will evaluate them on our dataset along with parameter tuning.

3.6.5 Clusters division

In stead of generating only two clusters in this binary classification task in which case only a few outliers are not included in the main cluster, we generate a set of clusters and make a binary division between clusters (Fig.3.9). Assuming that most of hosts are benign and safe we first sort the clusters by their sizes. Considering too few hosts as abnormal will result in high rate of detection miss while considering too many will raise the false alarm rate which will bother network administrators. In this task detecting abnormal hosts as much as possible is with higher priority. We pre-define $R\%$, the ratio of hosts classified in safe clusters, and use it to control number of clusters we need to handle. In Chapter 5 we determine the appropriate $R\%$ that maximize the detection performance.

3.7 XGBoost based misuse intrusion detection

In this section, We propose the LAN internal misuse intrusion detection strategy based on LAN communication feature extraction and XGBoost supervised learning algorithm with high accuracy and balanced precision and recall on practical network environment. Instead of setting

up simulation networks, our approach enables the direct application of proposed method in practical use in LAN without the gap between simulated networks and real networks.

Fig.3.10 demonstrates an overview of our proposed XGBoost based misuse intrusion detection system design. Pcap files collected from distributed LANs are stored in central database on cloud server. We extract features and labels as described in this chapter. Pre-processed dataset is split into train dataset for training, validation dataset for overfitting prevention and test dataset for evaluation. With the imbalance characteristic in train dataset, we apply over and under sampling techniques before model training. We discuss details of each step in following sub-sections.

3.7.1 Labeling

In simulated attack data collection method, attacking or simulated attacking tools are applied to create malicious communications in network, where labels of positive(malicious) samples and attack types are natural to be available in advance. In contrast, in practical network environment we collect the real attacks for training, raising the difficulty of positive sample labeling.

In our dataset, hosts are labelled as normal or malicious according to the existence of unicast packets sent towards monitoring devices. As monitoring devices passively react to coming communications, hosts who send TCP and UDP unicast packets are considered to be malicious because they recognize the MAC address of monitoring device and tries to communicate with it when receiving no active actions from the device. For TCP case, we label the host to be positive when TCP SYN packet is sent from it as this indicates that the host attempts to set up three-way handshake with monitoring device. For UDP case, we label the host as positive when it actively sends direct UDP unicast packet to monitoring device without receiving any packets from it.

It is necessary to mention that unicast packets directly sent from malicious hosts to monitoring host indicate that an attack has already started. Only detecting these unicast packets is too late for intrusion detection. In need to detect intrusions as early as possible, we apply machine learning strategy on features extracted from other protocols.

3.7.2 Pre-processing

Before feature matrix supervised model training we perform a scaling process on data matrix as to uniform the scales of each feature to be between the range of $[0, 1]$ by applying *MaxAbxScaler* algorithm. The scaled value for i -th sample on k -th feature x_i^k is described as:

$$x_i^{k'} = \frac{x_i^k}{\max_j |x_j^k|}, j = 1, 2, \dots, n \quad (3.1)$$

where n is the number of samples in feature matrix.

3.7.3 Over/under Sampling

Compared to simulated data collection where attacks are artificially introduced in networks, real network intrusion detection is faced with an inherent imbalanced classification problem. Like in almost all uneven data representation cases that the minority is usually the more important one [64], our intrusion detection also focus on the detection of malicious hosts who take a tiny

proportion in the dataset. The abundance of samples for majority class(negative class) swamp the minority class(positive class) in supervised learning model training.

One approach to handle imbalanced classification is to over sample the minority class by synthesizing new samples from the current samples. One famous strategy is a data augmentation method Synthetic Minority Oversampling Technique(SMOTE) [65]. SMOTE first randomly selects a minority sample and its k nearest minority neighbors. Then a neighbor is again randomly selected and these two samples form a line segment in the feature space, and synthetic samples are generated as a convex combination of these samples.

However, creating excessive minority samples to match the number of majority samples possibly result in ambiguous samples between classes especially when there is a strong overlap in the original data. Combined with over sampling technique, under sampling is often applied to reduce the data by eliminating samples belonging to the majority class [64]. A major drawback of under sampling is that it can discard potentially useful information inside the majority class.

In our approach we take advantage of the combination of over and under sampling technique. Specifically we use SMOTE for over sampling and RandomUnderSampler for under sampling, which randomly selects samples and removes from majority class.

3.7.4 Model training

With widely acknowledged efficiency and performance, XGBoost technique is frequently applied for a range of data mining and machine learning tasks [16]. As an optimized distributed gradient boosting library, XGBoost provides a parallel tree boosting that tackles problems of classification or regression. Compared to other supervised learning models including Naive Bayes, kNN and SVM, tree-based models are effective in handling data that are not normally distributed and easier in model explanation. Compared to decision tree based Random Forest and AdaBoost, XGBoost optimizes the loss function formed sequentially and introduced regularization terms for overfitting prevention.

In XGBoost, it contains a set of decision trees and in each step, a new tree is established based on dataset and feature set to optimize the target function. After the establishment of all the trees, prediction result of sample x_i can be described as:

$$\hat{y}_i = \sum_{k=1}^K f_k(x_i), f_k \in \mathcal{F} \quad (3.2)$$

where K is the number of decision trees and \mathcal{F} is the space of decision trees. In tree ensemble model of XGBoost, the loss function to be optimized at the t -th iteration(the step to establish the t -th tree) can be generalized as:

$$L^{(t)} = \sum_{i=1}^n l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i)) + \Omega(f_t) \quad (3.3)$$

where l is a differentiable convex loss function that measures the difference between the prediction \hat{y}_i and the true label y_i . $\Omega(f_t)$ penalizes the complexity of the model with regularization terms.

When adding the new tree function f_t , the loss function is transformed to the representation of first and second order gradient statistics. XGBoost selects the best new tree model to optimize

the loss function.

The wide use of XGBoost benefits from its efficiency resulting from parallel processing and performance due to the regularization term that prevents the model from overfitting. To further overcome the overfitting problem where model performs apparently better on train dataset than on test dataset, we apply early stopping method with validation dataset. The RMSE(Root Mean Square Error) describes the ratio of square deviation between true label y_i and predicted label \hat{y}_i among all n samples, whose formula is:

$$RMSE = \sqrt{\frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n}} \quad (3.4)$$

By monitoring RMSE value on validation set, we control training process not to be in overfitting. When validation RMSE does not decrease in consecutive T epochs, we stop the training so that it performs equally well on both training set and validation set.

Chapter 4

Implementation

4.1 LAN monitoring system implementation

For client side, we have used Raspberry Pi 3 platform as our data collection monitoring node (Fig.4.1). In the LAN network of our collaborators, we connect the monitoring on the normal port of a switch. On the monitoring node there are mainly 3 main functions: pcap file collection based on tcpdump, Cowrie Honeypot [22], and Dionaea Honeypot [20]. The data, including pcap file and honeypot logs, are sending from this client at midnight of local timezone, with a limitation of the bandwidth, restricting the influence of normal network use to the least level. For the server side, we set up data collection/processing server in our lab in the University of Tokyo.

4.2 Dataset creation

In this thesis our dataset is collected in 45 networks with one monitoring device in one network, as shown in Fig.4.2. Among them 28 devices are installed with honeypots while 17 are not. They are deployed in 10 countries. We are still planning to deploy more devices in other networks and other countries.

In this thesis we make our machine learning based detection based on data from Nov.1st, 2019 to May.5th, 2020, 187 days in total. We have increasing size of data because of real-time collection.



Figure 4.1: LAN Monitoring Node: Implemented on Raspberry Pi

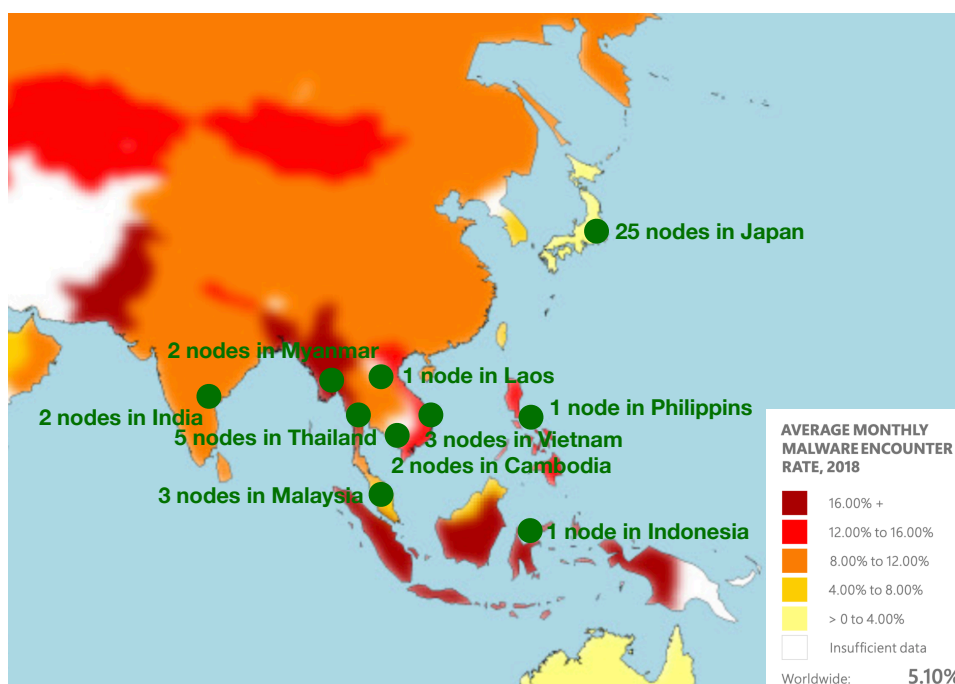


Figure 4.2: Number of monitoring devices deployed in different countries. Background: Average monthly malware encounter rate 2018, Microsoft security intelligence report [66]

Chapter 5

Experiment & Evaluation

5.1 Demonstration of 3-level detection

5.1.1 Experiment settings

We deployed one monitoring node(n001) on a LAN in our laboratory. Our server was also setup on our lab. From another host, we conducted two types of vulnerability tests with using nmap. We also simulated two types of real attacks that try to intrude into the monitoring node. We checked if our algorithm could differentiate these events appropriately. Our node was set to static IP address 172.16.1.231, while our attacker was set with 172.16.1.208.

Case 1 ARP Scan. We made ARP scan to 256 IP addresses within 172.16.1.0/24 with nmap command:

```
nmap -Pn 172.16.1.0/24
```

Case 2 TCP Port Scan. After ARP scan, We made TCP port scan to the IP address of the monitoring node by:

```
nmap -sT 172.16.1.231
```

Case 3 Application-Layer Connection Establishment. We used SMB client to connect with SMB service running on the monitoring node by Dionaea honeypot, and closed the connection. We repeated this several time. This simulates brute-force attacks made by malware.

Case 4 Downloading a File. We login to telnet running on the node with guessed username and password, and after successful login, we executed commands to generate a file on the honeypot's platform.

5.1.2 Case 1 and 2: Vulnerability tests

As for Case 1, the system has developed the following report.

Detected ARP scan from IP: 172.16.1.208 (MAC: a0:99:9b:1a:30:7f) on n001 It scans 256 IP addresses.

...
2019-04-15 16:38:40.175681 Who has 172.16.1.1 tell 172.16.1.208
2019-04-15 16:38:40.175683 Who has 172.16.1.2 tell 172.16.1.208
2019-04-15 16:38:40.177130 Who has 172.16.1.3 tell 172.16.1.208
2019-04-15 16:38:40.177133 Who has 172.16.1.4 tell 172.16.1.208
2019-04-15 16:38:40.177137 Who has 172.16.1.5 tell 172.16.1.208
2019-04-15 16:38:40.177139 Who has 172.16.1.6 tell 172.16.1.208
2019-04-15 16:38:40.177142 Who has 172.16.1.7 tell 172.16.1.208
...

This report indicates that there was a potential activities for discovering all the hosts in the network.

As for Case 2, the system has developed the following report.

Detected 1008 TCP SYN attacks from IP: 172.16.1.208 (MAC: a0:99:9b:1a:30:7f) during and after the ARP scan.

...
2019-04-15 16:41:22.088210 172.16.1.208:40668->172.16.1.231:135
2019-04-15 16:41:22.088470 172.16.1.208:41788->172.16.1.231:1720
2019-04-15 16:41:22.088810 172.16.1.208:38802->172.16.1.231:23
2019-04-15 16:41:22.089080 172.16.1.208:39808->172.16.1.231:443
2019-04-15 16:41:22.089320 172.16.1.208:60796->172.16.1.231:1025
2019-04-15 16:41:22.089565 172.16.1.208:43746->172.16.1.231:139
2019-04-15 16:41:22.089835 172.16.1.208:52368->172.16.1.231:8080
2019-04-15 16:41:22.090004 172.16.1.208:55746->172.16.1.231:587
2019-04-15 16:41:22.090156 172.16.1.208:55028->172.16.1.231:5900
2019-04-15 16:41:22.090400 172.16.1.208:35452->172.16.1.231:1723
2019-04-15 16:41:22.090593 172.16.1.208:58040->172.16.1.231:21
2019-04-15 16:41:22.090845 172.16.1.208:39818->172.16.1.231:445
...

This results indicate that only TCP scan activities were observed after the ARP scan, which were potentially just scans similar with nmap – vulnerability tester.

5.1.3 Case 3 and 4: Malware attacks

As for Case 3, the system has developed the following report.

This result shows that the host has established connections to the monitoring node with SMB protocol, which potentially be a malware attacks or an enhanced network scans [58].

As for Case 4, the system has developed the following report.

Timestamp	Dionaea Status	Source IP	Protocol
2019-04-15 17:04:25.112226	connection.tcp.accept	172.16.1.208	smbd
2019-04-15 17:04:27.774802	connection.free	172.16.1.208	smbd
2019-04-15 17:04:28.456049	connection.tcp.accept	172.16.1.208	smbd
2019-04-15 17:04:30.581191	connection.free	172.16.1.208	smbd
2019-04-15 17:04:31.068269	connection.tcp.accept	172.16.1.208	smbd
2019-04-15 17:04:33.179649	connection.free	172.16.1.208	smbd
2019-04-15 17:04:33.694936	connection.tcp.accept	172.16.1.208	smbd
2019-04-15 17:04:36.424633	connection.free	172.16.1.208	smbd

Timestamp	Cowrie Status	Source IP	Protocol	Details
2019-04-15 17:05:48.405094	session.connect	172.16.1.208	telnet	
2019-04-15 17:05:57.662858	login.success	172.16.1.208	telnet	user/pwd: "root/123"
2019-04-15 17:06:19.874716	command.input	172.16.1.208	telnet	cmd:"echo a >> t.txt"
2019-04-15 17:06:22.686089	file_download	172.16.1.208	telnet	url:"/home/root/t.txt"
2019-04-15 17:06:22.700734	log.closed	172.16.1.208	telnet	
2019-04-15 17:06:22.714036	session.closed	172.16.1.208	telnet	

This result shows that the malware has intruded into the honeypot area of the monitoring node and downloaded a file onto the platform. This really means that the malicious host really contained a real malware and attacked to the available nodes in the LAN.

5.2 Dataset host analysis

5.2.1 Observed hosts

In all the 45 nodes, we have observed totally 52,463 hosts identified by their MAC addresses. Number of observed hosts vary in different networks based on number of people using the network. In Fig.5.1 we show distribution in number of LANs for different number of hosts observed. Among 45 LANs, in 19 LANs 0-100 hosts are observed, taking 42.2% at the first place. In 9 LANs more than 600 hosts are observed and in 8 LANs 101-200 hosts are observed.

In the LAN where most hosts are observed we have 11,619 hosts, and for the least one we only observed 8 hosts. Averagely we observe 1,166 hosts by a monitoring device during the experimental period.

5.2.2 Unicast attacks

Among the 52,463 hosts, 14,228 hosts are collected in normal devices and 38,235 hosts are collected in honeypot-enabled devices. Table.5.1 shows number of confirmed malicious hosts (who have attacked monitoring device) in honeypot-enabled devices and normal devices, also classified in different attack types. We analyze number of hosts targeting on each TCP or UDP

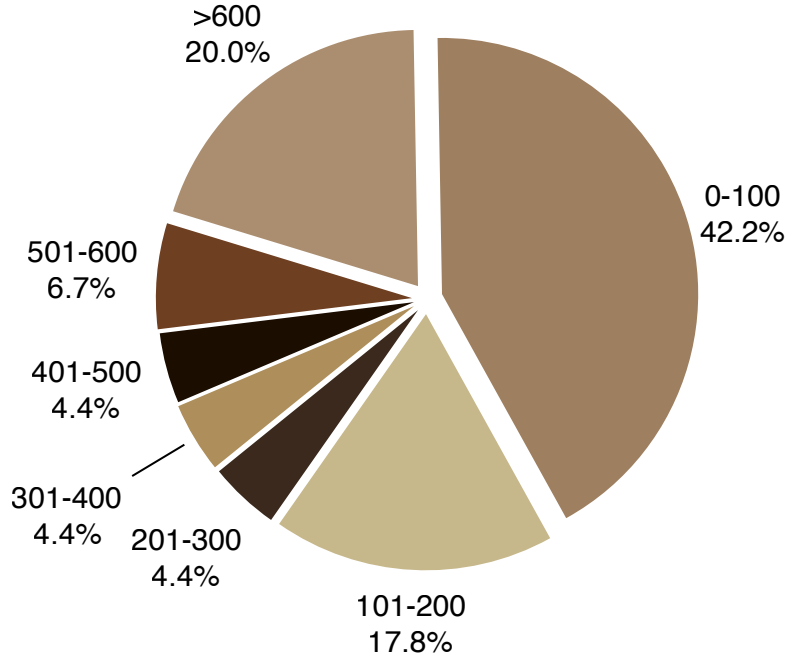


Figure 5.1: Distribution of LANs for each range of number of hosts. LANs where number of hosts observed is greater or equal to a and less than or equal to b is classified in $a - b$ sector.

Table 5.1: Number of confirmed malicious hosts

	NORMAL	HONEYPOT	Mixed
With TCP attacks	53	99	152
With UDP attacks	72	268	340
Mixed	106	315	421

port, shown in Fig.5.2. We only demonstrate TCP ports to which number of attackers is more than 24 and UDP ports to which number of attackers is more than 3 because port scanning of some hosts makes the diagram unclear if we show all.

TCP port 445 for SMB protocol suffers from attacks from 110 attackers (which is the most). Top 10 TCP ports are 445(SMB), 80(HTTP), 139(NetBIOS), 62078(UPnP, iTunes), 21(FTP), 443(HTTPS), 135(RPC), 3389(RDP), 8009(Netware HTTP), 631(Mac OS X Printer Sharing), 8080(HTTP).

UDP port observed to be accessed by most hosts(190 hosts) is port 137 for NBNS. Top 10 UDP ports are 137(NBNS), 21346(Unknown), 68(Bootstrap protocol client), 1900(SSDP), 161(SNMP), 5353(MDNS), 69(TFTP), 2054(Weblogin), 53(DNS), 7(Echo), 3702(WSD).

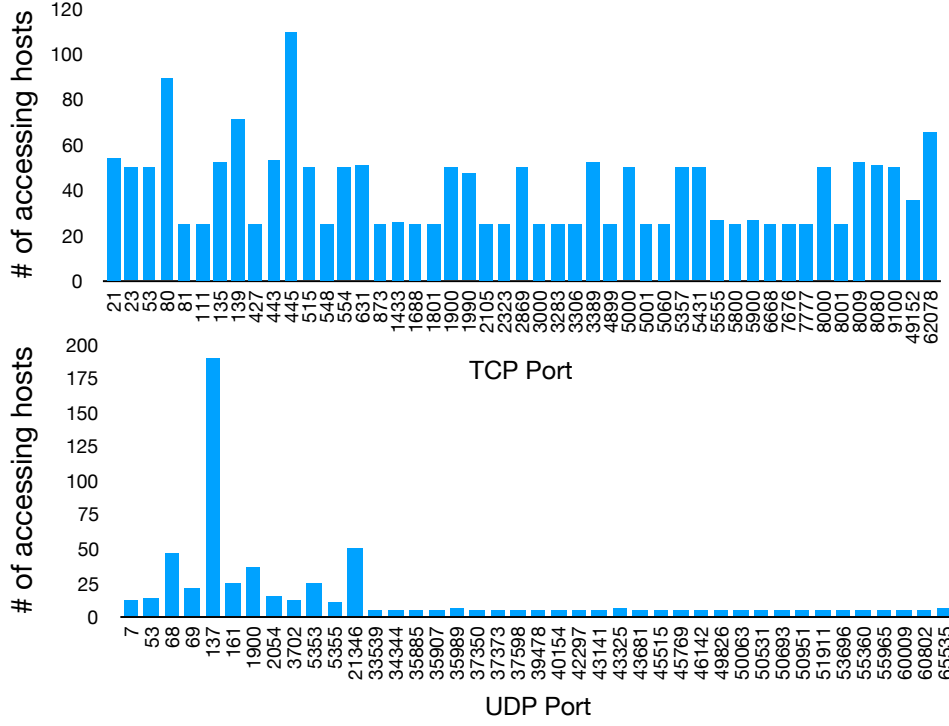


Figure 5.2: For each TCP or UDP port, we calculate number of hosts attacking on this port on monitoring device. We only show TCP ports with attacking hosts more than 24 and UDP ports with attacking hosts more than 3.

5.3 Evaluation of clustering based anomaly detection

5.3.1 Evaluation metrics of clustering-based detection

As described in Chapter 3, we evaluate how effective our detection method is based on the ratio of detected confirmed malicious hosts over all confirmed malicious hosts. Confirmed malicious hosts are validated for having sent TCP packets to monitoring device. If we can detect nearly all hosts that have been discovered to make attacks, we believe our detection method is useful in malicious host detection. In another word, we evaluate how our method could recall the confirmed malicious host:

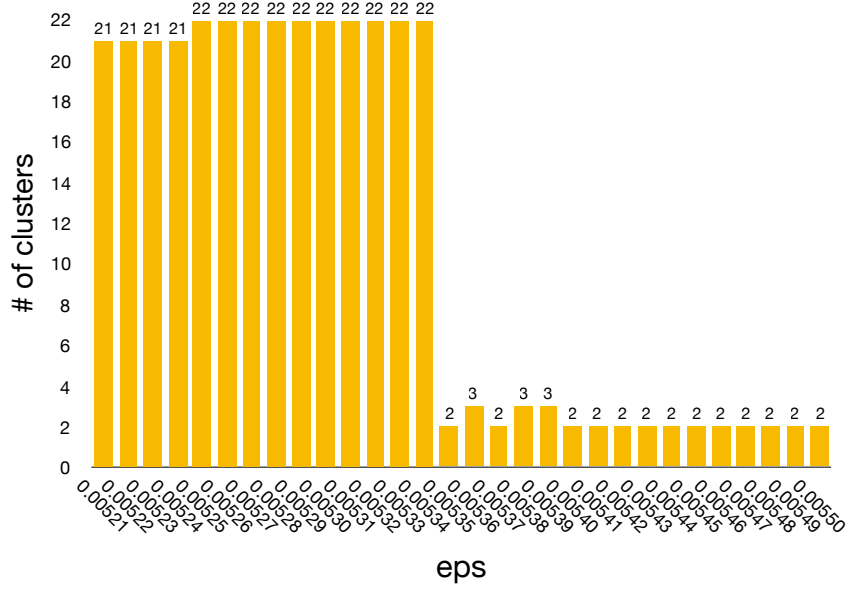
$$Malicious Recall = \frac{\# \text{ of detected confirmed malicious hosts}}{\# \text{ of all confirmed malicious hosts}}$$

To make a further distinction, we define *Malicious TCP Recall* for only confirmed malicious hosts who made TCP attacks to monitoring device, while *Malicious UDP Recall* for only those who made UDP attacks. We discuss performance separately in these two cases.

Also, we control the ratio of abnormal hosts by $R\% = 90\%$ in our experiments.

5.3.2 DBSCAN evaluation

As a density-based clustering method, DBSCAN is unique by unnecessary of pre-defined number of clusters. Nevertheless, DBSCAN clustering performance shows sensitivity on its two

Figure 5.3: DBSCAN: Number of clusters under min_points as 100 and different eps

main parameters: ϵ (or eps) and min_points . Especially for eps , if it is set too small then a big cluster will be decomposed into small ones, while if too large several clusters will not be differentiated. When clusters are with different densities, in DBSCAN it becomes impossible to find an eps that is available in clustering.

Based on our dataset we encountered exactly this specific problem, shown in Fig.5.3. There is a huge jump of number of clusters at a point of eps when there is a tiny change (less than 10^{-8} in finer-grained experiments). When number of clusters is 21 or 22, malicious recall is as low as 0.02, which means it can hardly detect any malicious host out by clustering.

When number of cluster is 2, Table.5.2 shows number of hosts and number of confirmed malicious hosts in each cluster. In the smaller cluster C_2 , it contains nearly all confirmed malicious hosts which is what we have expected, while it also contains too many hosts in total. It is unpractical that nearly 32% of hosts are believed to be malicious.

We cannot find an appropriate eps to differentiate safe clusters and abnormal clusters because of the densities of these two kinds of clusters are different. This demerit limits the use of DBSCAN on our clustering based detection.

Table 5.2: Number of hosts in each cluster for eps as 0.00536 in DBSCAN

	C_1	C_2	Total
# of hosts	34,598	17,865	52,463
# of confirmed malicious hosts	3	149	152
malicious recall	0.02	0.98	1

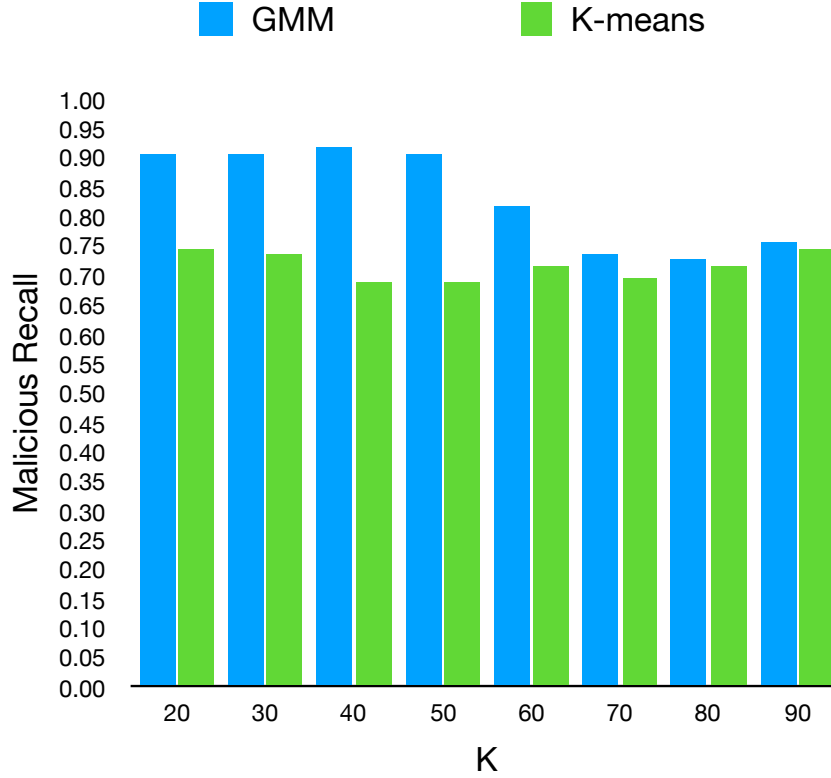


Figure 5.4: Malicious recall for mixed attack types collected in honeypot-enabled devices on GMM and K-means

5.3.3 K-means & GMM evaluation

In both K-means and GMM method, the only parameter to be tuned is the number of clusters K . Besides K we also evaluate performance on different monitoring devices: honeypot-enabled device and normal devices. We focus on malicious recall for three types: malicious TCP recall, malicious UDP recall and mixed malicious recall defined in Section 5.3.1.

We compare performance in both GMM and K-means in Fig.5.4, under the dataset collected in honeypot-enabled devices. GMM performs much better in recalling confirmed malicious hosts than K-means when K is under 60. According to our analysis on these two algorithms, an apparent demerit of K-means compared to GMM is that it requires the shape of each cluster to be convex and spherical. With no assumption on cluster shapes, GMM is more suitable in clustering tasks.

Fig.5.5 shows why GMM performs worse with increasing K value. Recalling confirmed malicious hosts who have made TCP attacks is not much influenced while for UDP case there exists a severe decline in performance. Also, we can be informed of the fact from this figure that generally predicting malicious hosts making UDP attacks is more difficult than predicting malicious hosts making TCP attacks, although on GMM it still reaches 90.7% as the recall rate compared to 96.0% for TCP. We also demonstrate that GMM performs almost the same on dataset collected from honeypot-enabled devices with normal devices in Fig.5.6. This eliminates the concerns that installed honeypot may affect the clustering-based detection. With honeypot logs we are capable of analyzing further on attack behaviors of malicious hosts.

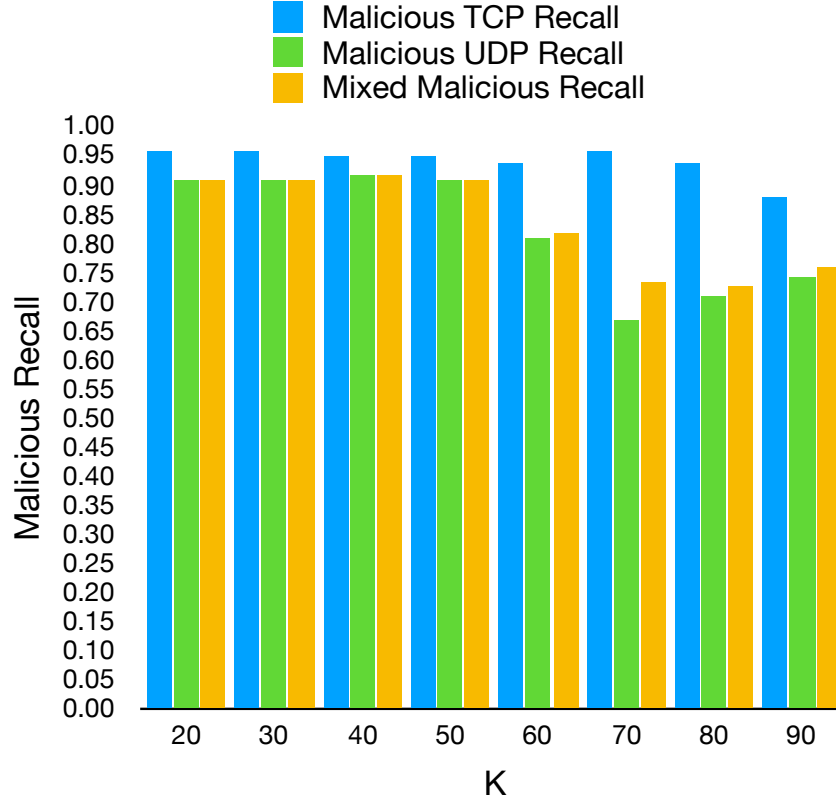


Figure 5.5: Malicious recall for different attack types collected in honeypot-enabled devices on GMM

In general, when we include clusters containing $R\%=90\%$ hosts as safe clusters, 96.0% of confirmed malicious hosts with TCP and 90.7% confirmed malicious hosts with UDP are detected. With this rate we can believe that other hosts in same clusters are potentially malicious.

5.3.4 False alarm rate controlling

As we have discussed in Chapter 3, by pre-define $R\%$ we include a ratio of hosts that belong to safe clusters. For example, when we set $R\%=90\%$, we need network administrators to check hosts clustered in last several clusters(sorted by size), and the sum of number of hosts in these clusters is around 10%. When we set the $R\%$ value larger we are able to detect more abnormal hosts but false alarm rate will increase.

Fig.5.7 demonstrates how $R\%$ affects performance. Both of malicious TCP recall and malicious UDP recall decreases when $R\%$ becomes larger as we have expected, and malicious UDP recall falls faster.

For an administrator, apparently it is better to sacrifice false alarm rate for better recall rate to detect malicious host as many as possible. Under the satisfaction on recall rate, we can control the last 9% of hosts as abnormal, according to our experiment result.

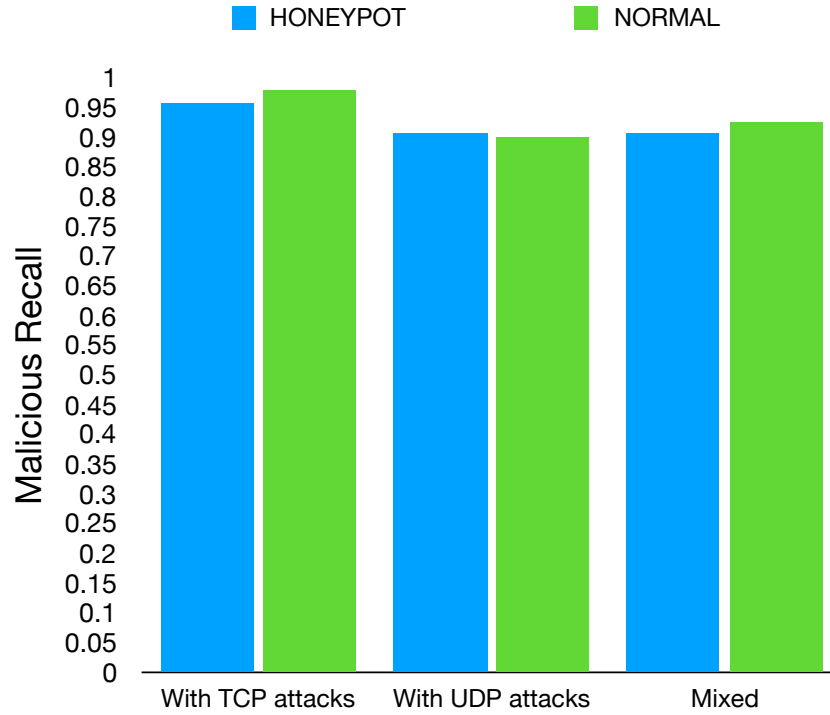


Figure 5.6: Malicious recall for different attack types collected in different type of devices on GMM

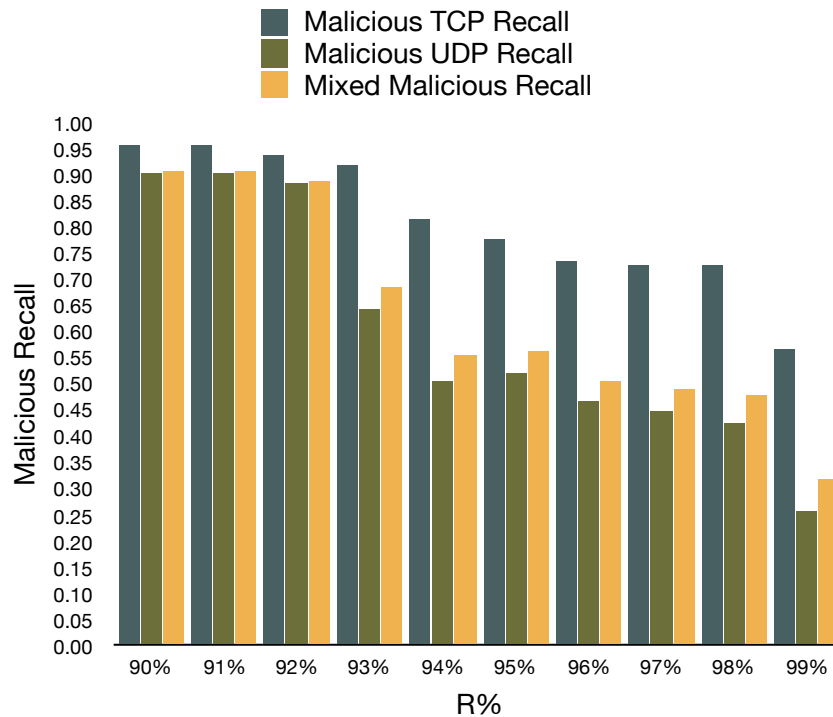


Figure 5.7: Malicious recall with different $R\%$ collected in honeypot-enabled devices on GMM

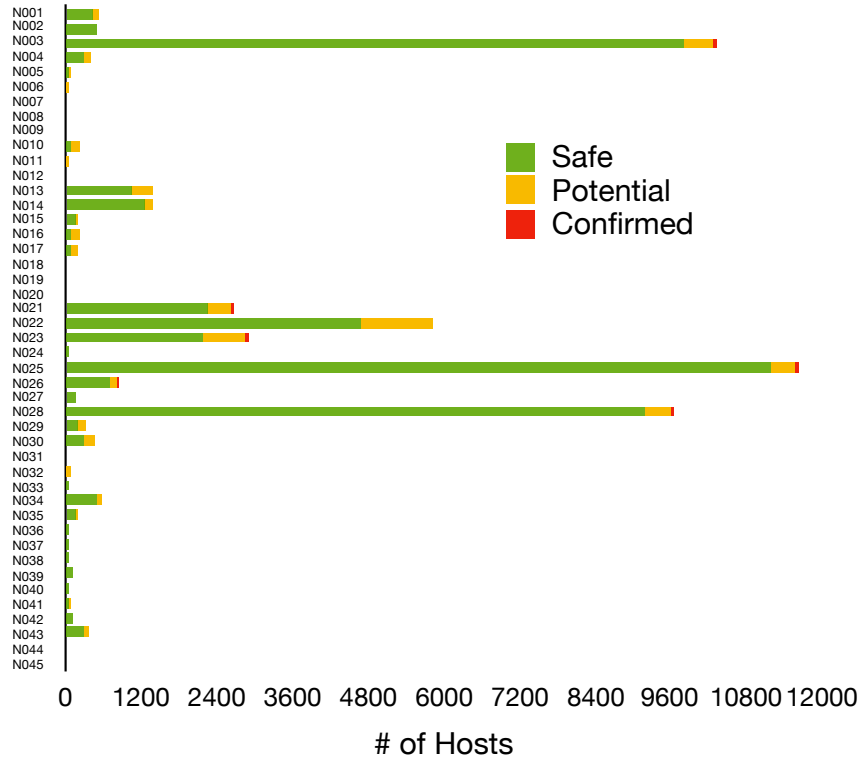


Figure 5.8: Safe, potentially malicious and confirmed malicious hosts in each monitoring device

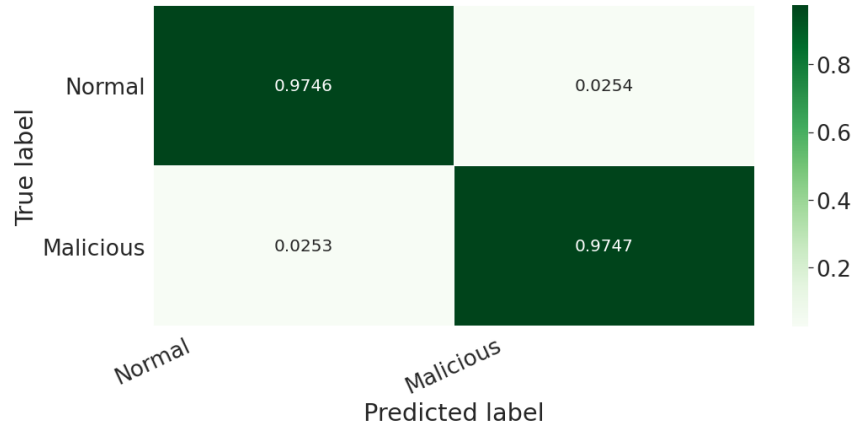


Figure 5.9: Confusion matrix of classification result with XGBoost

5.3.5 Detection result

During Nov.1st, 2019 to May.5th, 2020, we detect 421 confirmed malicious hosts and 5,677 abnormal hosts (including confirmed malicious hosts) in 45 networks. We illustrate our detection result on each monitoring device in Fig.5.8.

Table 5.3: Classification result on different algorithms

Model	Negative Class			Positive Class			Weighted Average		
	Precision	Recall	F1-score	Precision	Recall	F1-score	Precision	Recall	F1-score
BernoulliNB	0.828	0.728	0.775	0.479	0.624	0.542	0.729	0.698	0.709
5NN	0.878	0.966	0.920	0.886	0.667	0.761	0.881	0.880	0.874
SVM	0.961	0.853	0.904	0.714	0.914	0.802	0.891	0.871	0.875
AdaBoost	0.970	0.961	0.965	0.905	0.925	0.915	0.951	0.951	0.951
Random Forest	0.986	0.922	0.953	0.833	0.968	0.896	0.942	0.935	0.937
XGBoost	0.990	0.975	0.982	0.939	0.975	0.957	0.975	0.975	0.975

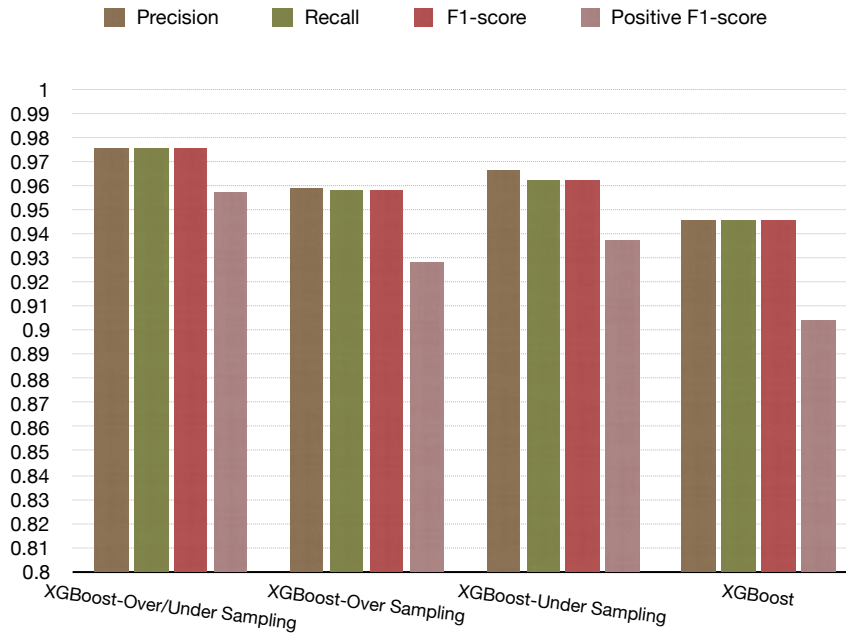


Figure 5.10: Model performance with different sets of sampling strategies

5.4 Evaluation of XGBoost based misuse detection

5.4.1 Experiment settings

In the experiment we set the ratio of NBNS and LLMNR abnormal query name threshold to be 0.3, indicating that if the ratio of vowel is larger than 0.3 then the query is considered to be abnormal. For early stopping in model training, if validation RMSE does not decrease for consecutive $T = 15$ then training process should be terminated. For over and under sampling on train data, we first perform SMOTE to add minority samples to 0.7 of majority samples, then perform RandomUnderSampler to remove majority samples so the ratio becomes 0.9. In our test dataset ratio of majority over minority is around 2.5.

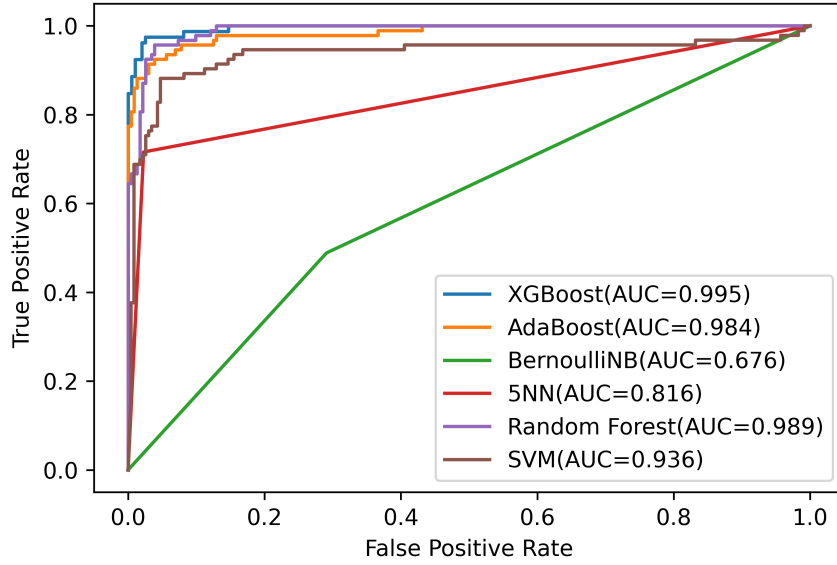


Figure 5.11: ROC curve on different algorithms

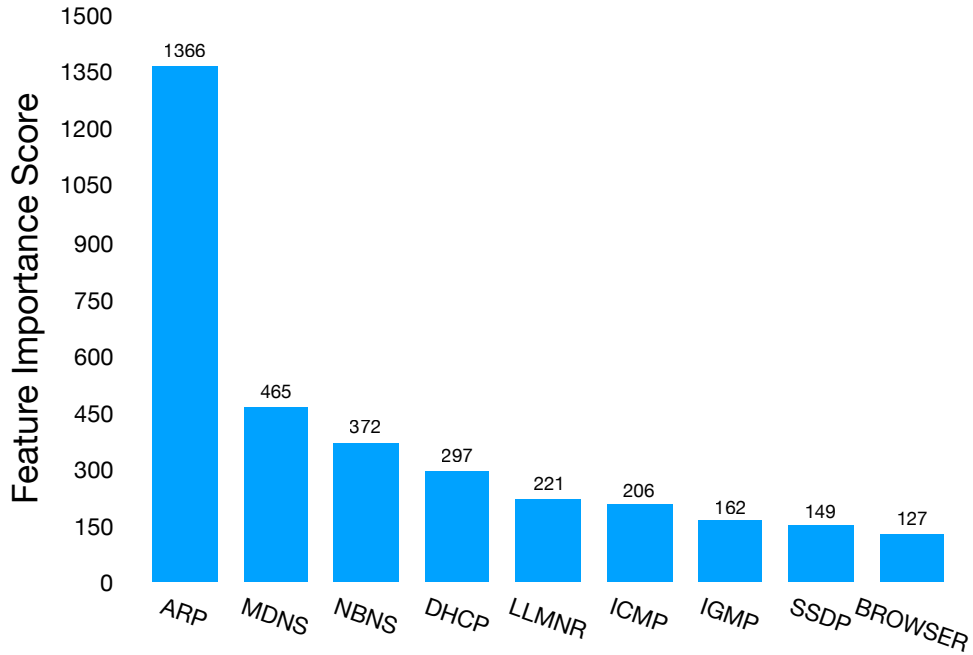


Figure 5.12: Sum of Feature Importance Score for each Protocol

5.4.2 Classification result

We first provide overall classification result on test data. On handling an intrusion detection task, we value more on recall rate of the positive class compared to FPR(false positive rate). Overall accuracy reaches 0.975 with precision rate on positive samples at 0.939 and recall rate on positive samples at 0.975, resulting in a 0.957 f1-score. Confusion matrix described in Fig.5.9

indicates that our model performs equally well on safe hosts and malicious hosts classification.

5.4.3 Effectiveness of over/under sampling

Fig.5.10 demonstrates the effectiveness of applying over and under sampling techniques on train data. Either of them enhances the overall performance (precision, recall, f1-score and f1-score of positive class) of the model, and the combination of these two techniques promote the performance to our best level. As analyzed in Section 3.7.3, both of the sampling techniques handle the imbalanced data problem. Nevertheless, they both introduce risks in training: SMOTE may make the edge of the classes ambiguous and RandomUnderSampler cause loss of information, which limit us from using only one of them to reach the balanced ratio directly.

5.4.4 Comparison with other models

We compared the performance of XGBoost classifier with other widely used machine learning models: Bernoulli Naive Bayes, 5NN(K=5 on KNN), SVM, AdaBoost and Random Forest. Fig.5.11 shows the ROC curve and AUC(Area Under the Curve) value to compare their performance. The three ensemble learning algorithms: Random Forest, AdaBoost and XGBoost perform competitively well. Compared to Random Forest who selects random samples and random features when establishing each decision tree, new tree generation in XGBoost in each step is to optimize the loss function formed sequentially. Compared to AdaBoost, XGBoost introduced regularization terms for overfitting prevention. These differences make XGBoost perform best on our intrusion detection task. Precision, recall and f1-score on each class is shown in Table 5.3.

5.4.5 Feature importance

Besides classification result, we can also attain the feature importance score generated by XGBoost model. Feature importance score for each decision tree is the amount of the improvement of the performance measure used to select the split on each tree node. Then the importance for each feature is calculated by the average of importance for each tree. It reflects the significance of each feature in influencing the classification result.

We aggregate sum of feature importance scores of each feature by the protocol they belong to in Fig.5.12. ARP significantly influences the effectiveness of LAN intrusion detection, which is the result we could predict as malicious hosts usually need to conduct ARP scanning for MAC addresses of other hosts. Following ARP, MDNS, NBNS, DHCP, LLMNR and ICMP can also be considered important in LAN intrusion detection. The feature importance score can also be used to reduce feature space dimension in the process of feature extraction, which is beyond the scope of this paper and considered to be included in the future work.

Chapter 6

Conclusion

6.1 Conclusion

We proposed an architecture of cloud-based LAN-security monitoring system that can differentiate vulnerability tests and malware attacks happened in the remote local area networks. We presented our design of monitoring node with honeypot installed. We designed the algorithm run on the collection server to recognize abnormal activities such as ARP scan, TCP port scan, application-level connection establishment and intrusion. We demonstrated that our system could recognize those behaviors, pointing out that latter two activities are mostly caused by malware not by vulnerability testing.

This thesis also proposes real-world and real-time LAN traffic dataset collection method and anomaly & misuse based intrusion detection based on it. We extracted features of hosts based on a range of protocols specifically designed for LANs. We analyzed three clustering methods: K-means, DBSCAN and GMM for clusters training. We also proposed misuse detection on LAN targeted intrusions with XGBoost. Detection model with XGBoost is trained with over/under sampling techniques for imbalanced classification problem and overfitting prevention strategy.

Our dataset contains traffic data of 52,463 observed hosts during Nov.1st, 2019 to May.5th, 2020 collected from 45 networks in 10 countries within the scope of this paper. Evaluation based on this dataset validate the effectiveness of our proposed detection method in achieving high rate of detecting confirmed malicious hosts (96.0% for confirmed TCP attacking hosts and 90.7% for confirmed UDP attacking hosts), enabling us to believe in the abnormality of other detected hosts (potentially malicious hosts). It also demonstrates that our misuse detection performs 97.5% in overall precision and 97.5% in overall recall. Besides, we also discovered that ARP, MDNS and NBNS are the top 3 protocols that influence the intrusion detection most in LAN.

6.2 Discussion & future Works

Importance of distinguishing vulnerability tests from malicious activities should not be ignored because tests for network status, host existence and ports status are not always conducted by administrators. Each host with security tools or software has potential for conducting a test to check if current LAN is secure enough for itself to connect in, which has been proved by our

discovery on several security tools who make ARP scans and check TCP ports periodically.

In the scope of this thesis, we do not consider on IP spoofing, but it could really happen in the real network. Also there are cases when malicious hosts are continuously changing its IP addresses. With the expansion of experiment range, we will include considerations in future work.

In the clustering based anomaly detection we use the whole dataset collected from a number of monitoring devices(corresponding to a network separately). We can see from Fig.5.8 that in some of the networks (whose device labeled as N006, N032, etc.) most of the hosts are detected to be potentially malicious. This is mainly because they contain only a few hosts in network and if those hosts behave similarly, they tend to be classified in the same cluster. Before clustering hosts, in the next step we consider classifying networks in groups with a smart method and conduct clustering based detection for each group.

Nowadays, based on the aggregation of sensors and information input interface carried on smart devices, unprecedented amount of data is available for applications. The sensitivity nature of the data means the risks to store the data in a centralized location, which is the current situation for most applications. With this consideration, a learning technique that allows users to train the model from the rich data, without the need to centrally store it is proposed as Federated Learning [67–69].

As we have discussed, network data flow collected in LAN also contains a relatively high level of sensitivity. And we spontaneously have distributed system: each monitoring device could be a client for training. Federated learning could solve the challenge appropriately. In the future work on solving these problems, we believe that this will be an innovative work of using federated learning to detect intrusion detection in LAN.

List of Publications

- International Conferences(Peer-reviewed)
 - Zhiqing Zhang, Hiroshi Esaki, and Hideya Ochiai. "Analysis of Malware Hidden Behind Firewalls with Back Scans." 2019 7th International Symposium on Digital Forensics and Security (ISDFS). IEEE, 2019.
 - Zhiqing Zhang, Hiroshi Esaki, and Hideya Ochiai. "Unveiling Malicious Activities in LAN with Honeypot." 2019 4th International Conference on Information Technology (InCIT). IEEE, 2019.
 - Zhiqing Zhang, Hiroshi Esaki, and Hideya Ochiai. "XGBoosted Misuse Detection in LAN-Internal Traffic Dataset" 2020 18th International Conference on Intelligence and Security Informatics (ISI). IEEE, 2020.
 - Kobayashi Hyuga, Zhiqing Zhang, Hideya Ochiai, and Hiroshi Esaki. "Probing Firewalls of Malware-Infected Networks with Honeypot." Proceedings of the 14th International Conference on Future Internet Technologies. ACM, 2019.
 - Ying Luo, Zhiqing Zhang, Hiroshi Esaki, and Hideya Ochiai. "Classification of TCP 445 Attacks and Global Snapshot with Honeypot Analysis." 2019 International Conference on Advanced Information Technologies (ICAIT). IEEE, 2019.

Bibliography

- [1] Neminath Hubballi, S Roopa, Ritesh Ratti, Ferdous A Barbhuiya, Santosh Biswas, Arijit Sur, Sukumar Nandi, and Vivek Ramachandran. An active intrusion detection system for lan specific attacks. In *Advances in Computer Science and Information Technology*, pages 129–142. Springer, 2010.
- [2] Mohammed Nadir Bin Ali, Mohamed Emran Hossain, and Md Masud Parvez. Design and implementation of a secure campus network. *International Journal of Emerging Technology and Advanced Engineering*, 5(7):370–374, 2015.
- [3] Gordon Fyodor Lyon. *Nmap network scanning: The official Nmap project guide to network discovery and security scanning*. Insecure, None, 2009.
- [4] Vinod Kumar and Om Prakash Sangwan. Signature based intrusion detection system using snort. *International Journal of Computer Applications & Information Technology*, 1(3):35–41, 2012.
- [5] Neminath Hubballi and Vinoth Suryanarayanan. False alarm minimization techniques in signature-based intrusion detection systems: A survey. *Computer Communications*, 49:1–17, 2014.
- [6] VVRPV Jyothsna, VV Rama Prasad, and K Munivara Prasad. A review of anomaly based intrusion detection systems. *International Journal of Computer Applications*, 28(7):26–35, 2011.
- [7] Anna L Buczak and Erhan Guven. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications surveys & tutorials*, 18(2):1153–1176, 2015.
- [8] Maryam M Najafabadi, Taghi M Khoshgoftaar, Clifford Kemp, Naeem Seliya, and Richard Zuech. Machine learning for detecting brute force attacks at the network level. In *2014 IEEE International Conference on Bioinformatics and Bioengineering*, pages 379–385. IEEE, 2014.
- [9] Jiaqi Li, Zhifeng Zhao, Rongpeng Li, and Honggang Zhang. Ai-based two-stage intrusion detection for software defined iot networks. *IEEE Internet of Things Journal*, 6(2):2093–2102, 2018.
- [10] Chuanlong Yin, Yuefei Zhu, Jinlong Fei, and Xinzheng He. A deep learning approach for intrusion detection using recurrent neural networks. *Ieee Access*, 5:21954–21961, 2017.

- [11] Alex Shenfield, David Day, and Aladdin Ayesh. Intelligent intrusion detection systems using artificial neural networks. *ICT Express*, 4(2):95–99, 2018.
- [12] Dmitry V. Pantiukhin. Intelligent methods for intrusion detection in local area networks. 2019.
- [13] Lin Zhang, Meng Li, Xiaoming Wang, and Yan Huang. An improved network intrusion detection based on deep neural network. In *IOP Conference Series: Materials Science and Engineering*, volume 563, page 052019. IOP Publishing, 2019.
- [14] Ahmad Javaid, Quamar Niyaz, Weiqing Sun, and Mansoor Alam. A deep learning approach for network intrusion detection system. In *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIO-NETICS)*, pages 21–26. ICST, 2016.
- [15] Nour Moustafa, Gideon Creech, Elena Sitnikova, and Marwa Keshk. Collaborative anomaly detection framework for handling big data of cloud computing. In *2017 Military Communications and Information Systems Conference (MilCIS)*, pages 1–6. IEEE, 2017.
- [16] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794, 2016.
- [17] KDD Cup. Data (1999). URL <http://www.kdd.org/kdd-cup/view/kdd-cup-1999/Data>, 1999.
- [18] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. A detailed analysis of the kdd cup 99 data set. In *2009 IEEE symposium on computational intelligence for security and defense applications*, pages 1–6. IEEE, 2009.
- [19] Marcin Nawrocki, Matthias Wählisch, Thomas C Schmidt, Christian Keil, and Jochen Schönfelder. A survey on honeypot software and data analysis. *arXiv preprint arXiv:1608.06249*, 2016.
- [20] Dionaea documentation release 0.8.0. 2018.
- [21] Tillmann Werner. Honeytrap-a dynamic meta-honeypot daemon, 2009.
- [22] Michel Oosterhof. Cowrie honeypot. *Security Intelligence*, 2014.
- [23] Yin Minn Pa Pa, Shogo Suzuki, Katsunari Yoshioka, Tsutomu Matsumoto, Takahiro Kasama, and Christian Rossow. Iotpot: analysing the rise of iot compromises. In *9th {USENIX} Workshop on Offensive Technologies ({WOOT} 15)*, 2015.
- [24] Georgios Portokalidis, Asia Slowinska, and Herbert Bos. Argos: an emulator for fingerprinting zero-day attacks for advertised honeypots with automatic signature generation. *ACM SIGOPS Operating Systems Review*, 40(4):15–27, 2006.
- [25] Muhammet Baykara and Resul Daş. A survey on potential applications of honeypot technology in intrusion detection systems. *International Journal of Computer Networks and Applications (IJCNA)*, 2(5):203–208, 2015.

- [26] Rohit Shukla and Maninder Singh. Pythonhoneymonkey: Detecting malicious web urls on client side honeypot systems. In *Proceedings of 3rd International Conference on Reliability, Infocom Technologies and Optimization*, pages 1–5. IEEE, 2014.
- [27] Kartik Chawda and Ankit D Patel. Dynamic & hybrid honeypot model for scalable network monitoring. In *International conference on information communication and embedded systems (ICICES2014)*, pages 1–5. IEEE, 2014.
- [28] Pagar Harshali Yashwant, Pathare Anjali Sanjay, and Shaikh Sameer Shekhanur. Buckler: Intrusion detection and prevention using honeypot.
- [29] Neha Agrawal and Shashikala Tapaswi. The performance analysis of honeypot based intrusion detection system for wireless network. *International Journal of Wireless Information Networks*, 24(1):14–26, 2017.
- [30] Muhammet Baykara and Resul Das. A novel honeypot based security approach for real-time intrusion detection and prevention systems. *Journal of Information Security and Applications*, 41:103–116, 2018.
- [31] Li Li, Hua Sun, and Zhenyu Zhang. The research and design of honeypot system applied in the lan security. In *2011 IEEE 2nd International Conference on Software Engineering and Service Science*, pages 360–363. IEEE, 2011.
- [32] Lionel Metongnon and Ramin Sadre. Beyond telnet: Prevalence of iot protocols in telescope and honeypot measurements. In *Proceedings of the 2018 Workshop on Traffic Measurements for Cybersecurity*, pages 21–26. ACM, 2018.
- [33] Mirosław Skrzewski. Monitoring malware activity on the lan network. In *International Conference on Computer Networks*, pages 253–262. Springer, 2010.
- [34] Zakir Durumeric, David Adrian, Ariana Mirian, Michael Bailey, and J Alex Halderman. A search engine backed by internet-wide scanning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 542–553, 2015.
- [35] Shehar Bano, Philipp Richter, Mobin Javed, Srikanth Sundaresan, Zakir Durumeric, Steven J Murdoch, Richard Mortier, and Vern Paxson. Scanning the internet for liveness. *ACM SIGCOMM Computer Communication Review*, 48(2):2–9, 2018.
- [36] Manos Antonakakis, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, J Alex Halderman, Luca Invernizzi, Michalis Kallitsis, et al. Understanding the mirai botnet. In *USENIX Security Symposium*, pages 1092–1110, 2017.
- [37] Zakir Durumeric, Eric Wustrow, and J Alex Halderman. Zmap: Fast internet-wide scanning and its security applications. In *USENIX Security Symposium*, volume 8, pages 47–53, 2013.
- [38] Fabrice J Ryba, Matthew Orlinski, Matthias Wählisch, Christian Rossow, and Thomas C Schmidt. Amplification and drdos attack defense—a survey and new perspectives. *arXiv preprint arXiv:1505.07892*, 2015.

- [39] Pitchai MohanaPriya, V Akilandeswari, G Akilarasu, and S Mercy Shalinie. An integrated approach of e-red and ant classification methods for drdos attacks. In *International Symposium on Security in Computing and Communication*, pages 304–312. Springer, 2014.
- [40] Hiroshi Tsunoda, Kohei Ohta, Atsunori Yamamoto, Nirwan Ansari, Yuji Waizumi, and Yoshiaki Nemoto. Detecting drdos attacks by a simple response packet confirmation mechanism. *Computer Communications*, 31(14):3299–3306, 2008.
- [41] Shyava Tripathi and Rishi Kumar. Raspberry pi as an intrusion detection system, a honeypot and a packet analyzer. In *2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS)*, pages 80–85. IEEE, 2018.
- [42] Asmaa Elsaedy, Kumudu S Munasinghe, Dharmendra Sharma, and Abbas Jamalipour. Intrusion detection in smart cities using restricted boltzmann machines. *Journal of Network and Computer Applications*, 135:76–83, 2019.
- [43] Sara A Althubiti, Eric Marcell Jones, and Kaushik Roy. Lstm for anomaly-based network intrusion detection. In *2018 28th International Telecommunication Networks and Applications Conference (ITNAC)*, pages 1–3. IEEE, 2018.
- [44] Hung Nguyen Viet, Quan Nguyen Van, Linh Le Thi Trang, and Shone Nathan. Using deep learning model for network scanning detection. In *Proceedings of the 4th International Conference on Frontiers of Educational Technologies*, pages 117–121. ACM, 2018.
- [45] Robin Sommer and Vern Paxson. Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE symposium on security and privacy*, pages 305–316. IEEE, 2010.
- [46] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.
- [47] Richard O Duda, Peter E Hart, and David G Stork. Pattern classification 2nd edition. *New York, USA: John Wiley&Sons*, 2001.
- [48] Paul Graham. A plan for spam, 2002. Available from World Wide Web: <http://www.paulgraham.com/spam.html>, 2003.
- [49] Stefan Axelsson. The base-rate fallacy and its implications for the difficulty of intrusion detection. In *Proceedings of the 6th ACM Conference on Computer and Communications Security*, pages 1–7. ACM, 1999.
- [50] Walter Willinger, Murad S Taqqu, Robert Sherman, and Daniel V Wilson. Self-similarity through high-variability: statistical analysis of ethernet lan traffic at the source level. *IEEE/ACM Transactions on networking*, 5(1):71–86, 1997.
- [51] Anja Feldmann, Anna C Gilbert, and Walter Willinger. Data networks as cascades: Investigating the multifractal nature of internet wan traffic. In *ACM SIGCOMM Computer Communication Review*, volume 28, pages 42–55. ACM, 1998.

-
- [52] Nour Moustafa and Jill 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)*, pages 1–6. IEEE, 2015.
 - [53] Jiong Zhang and Mohammad Zulkernine. Network intrusion detection using random forests. In *Pst. Citeseer*, 2005.
 - [54] Mehrnaz Mazini, Babak Shirazi, and Iraj Mahdavi. Anomaly network-based intrusion detection system using a reliable hybrid artificial bee colony and adaboost algorithms. *Journal of King Saud University-Computer and Information Sciences*, 31(4):541–553, 2019.
 - [55] Arif Yulianto, Parman Sukarno, and Novian Anggis Suwastika. Improving adaboost-based intrusion detection system (ids) performance on cic ids 2017 dataset. In *Journal of Physics: Conference Series*, volume 1192, page 012018. IOP Publishing, 2019.
 - [56] Sukhpreet Singh Dhaliwal, Abdullah-Al Nahid, and Robert Abbas. Effective intrusion detection system using xgboost. *Information*, 9(7):149, 2018.
 - [57] Sweta Bhattacharya, Rajesh Kaluri, Saurabh Singh, Mamoun Alazab, Usman Tariq, et al. A novel pca-firefly based xgboost classification model for intrusion detection in networks using gpu. *Electronics*, 9(2):219, 2020.
 - [58] S.Limjitti, H.Ochiai, H.Esaki, and K.Sripanidkulchai. Iot-vullock: Locking iot device vulnerability with enhanced network scans. In *the 13th International Conference on Ubiquitous Information Management and Communication (IMCOM 2019)*, 2019.
 - [59] Matsufuji Kai, S.Kobayashi, H.Esaki, and H.Ochiai. Arp request trend fitting for detecting malicious activity in lan. In *the 13th International Conference on Ubiquitous Information Management and Communication (IMCOM 2019)*, 2019.
 - [60] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.
 - [61] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.
 - [62] Chris Fraley and Adrian E Raftery. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American statistical Association*, 97(458):611–631, 2002.
 - [63] Douglas A Reynolds. Gaussian mixture models. *Encyclopedia of biometrics*, 741, 2009.
 - [64] Bartosz Krawczyk. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, 5(4):221–232, 2016.
 - [65] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

- [66] Microsoft security intelligence report 2018. *Microsoft Secur. Intell. Rep.*, 2018.
- [67] Brendan McMahan and Daniel Ramage. Federated learning: Collaborative machine learning without centralized training data. *Google Research Blog*, 3, 2017.
- [68] Hankz Hankui Zhuo, Wenfeng Feng, Qian Xu, Qiang Yang, and Yufeng Lin. Federated reinforcement learning. *arXiv preprint arXiv:1901.08277*, 2019.
- [69] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):12, 2019.

Acknowledgments

I would first express my sincere gratitude to my supervisor, Associate professor Hideya Ochiai, for all the guidance his gave on my research. With his assistance, I attained the precious opportunity to work in his laboratory and to continue my research topic on Internet-of-Things. With his supervision, I overcome the challenges in research and achieved my research goal during my Master's course. I can never forget those days hearkening his brilliant ideas, joining meaningful discussions and working in field work together with him in oversea visits. I would also express my gratitude to Professor Hiroshi Esaki and Associate professor Manabu Tsukada for their guidance on my research and presentations. I truly appreciate the discussions I have had with my lab members Ying Luo, Karakate Meatasit, Pawissakan Chirupphapa, Hyuga Kobayashi and Yuwei Sun during my research as they provided me with precious technical advices and ideas. I would like to thank all of our lab members, especially Ying Luo, Karakate Meatasit and Pawissakan Chirupphapa for making my lab life joyful. I would like to say thanks to secretaries, Aiko Iwai and Fumi Takahashi for assisting my administrative tasks and making our life in the laboratory comfortable. Outside lab, I would like to express my thankfulness to Xiaoyu Shang and my friend Sizhang Dong for their encouragement and assistance during these two years. Lastly, my heartfelt appreciation goes to my family for their support on my student life.

