AN UNSUPERVISED LEARNING APPROACH FOR NETWORK AND SYSTEM ANALYSIS

by

Duc Cong Le

Submitted in partial fulfillment of the requirements
for the degree of Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
March 2017

This thesis is dedicated to my parents

Table of Contents

List of Tables

List of Figures

Abstract

This thesis investigates the capability of employing the SOM, an unsupervised learning technique as a network data analytics system. In doing so, the aim is to understand how far such an approach could be pushed to analyze the network traffic, and to detect malicious behaviours. To this end, three different unsupervised SOM training schemes for different data acquisition conditions are employed. The approach is tested against publicly available botnet and malicious web request data sets. The results show that SOMs possess high potential as a data analytics tool on unknown traffic, and unseen attack behaviours. They can identify the botnet and normal flows with high confidence approximately 99% of the time on the data sets employed in this thesis, which is comparative to that of popular supervised and unsupervised learning methods in the literature. Furthermore, it provides unique visualization capabilities for enabling a simple yet effective network data analytic system.

List of Abbreviations and Symbols Used

Abbreviations

BMU Best Matching Unit

C&C Command and Control

CDR Class-wise Detection Rate

CRLF Carriage Return Line Feed

CSIC Spanish Research National Council

DDoS Distribution Denial of Service

DNS Domain Name System

DR Detection Rate

EM Expectation Maximization

HTTP HyperText Transfer Protocol

HTTPS HTTP Secure

ICMP Internet Control Message Protocol

IDS Intrusion Detection System

IoT Internet of Things

IP Internet Protocol

IRC Internet Relay Chat

LBNL Lawrence Berkeley National Laboratory

NIDS	Network Intrusion Detection System
P2P	Peer-to-Peer
PCA	Principle Component Analysis
ROC	Receiver Operating Characteristic
RTP	Real-time Transport Protocol
SOM	Self Organizing Map
SQL	Structured Query Language
SVM	Support Vector Machine
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
URL	Uniform Resource Locator
XSS	Cross Site Scripting

Symbols

c the index of the winning node / best matching unit

l SOM training length (number of iterations)

$M_1 \times M_2$ the size/number of neurons of a SOM, where M_1 and M_2 are two side lengths of the SOM lattice

N number of training instances

n number of dimensions of input space

$\sigma(t)$ the kernel width / neighbourhood radius at time t

τ the threshold for identifying important SOM nodes

$\mathcal{W}$ SOM lattice of neurons w_i

w_i weight vector of a SOM neuron/node/unit, $w_i = [\omega_{i1}, \omega_{i2}, ..., \omega_{in}]^T \in \mathbb{R}^n$

$\mathcal{X}$ list of all input vectors x

x an input vector, $x = [\xi_1, \xi_2, ..., \xi_n]^T \in \mathbb{R}^n$

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisors, Dr. Nur Zincir-Heywood and Dr. Malcolm I. Heywood, for their support, guidance, encouragement, and great ideas that have helped me complete the research project and this thesis. I would also like to thank Nova Scotia Government, Dalhousie University, and Mitacs for their supports toward this thesis research. Last but not least, I would like to acknowledge all the small things that helped me go through the life of a graduate student, from the things that kept distracting me from what I should be doing: Naruto (Masashi Kishimoto sensei) and animes of Ghibli studio to the things that helped me gain back the much needed focus: instrumental music, coffee, and tea.

Chapter 1

Introduction

The general utility of the Internet continues to grow on a yearly basis, and at the same time, so does the cybercrime threat landscape. There is a wide variety of network threats on the Internet, with different aims and attack vectors. Among these, botnet - the main focus of this thesis - has become one of the most dangerous threats [29][45]. Botnets consist of compromised machines, or bots, controlled by attackers (the botmasters) through Command and Control (C&C) communication channels. Botnets are responsible for many types of attacks these days, including but not limited to spam spreading, Distribution Denial of Service (DDoS) attacks, distribution of malicious software, information harvesting, identity theft, or exploiting victims' computational and network resources.

The threats are becoming more and more serious, as we are seeing the incoming wave of Internet of Things (IoT), which will connect a plethora of device categories - not only conventional computers or smartphones, but also smaller and lower cost devices, e.g. home appliances, security systems, portable medical devices, etc. Poorly protected IoT devices can be easily turned into a platform for attacking anything from individual websites to core parts of the Internet's infrastructure, hence opens the door for an once unthinkable generation of botnets. Just recently, in October 2016, the record of terabit-per-second DDoS attack was first achieved by an army of some 145,000 tiny compromised cameras, digital video recorders in two botnets - Mirai and Bashlight [17]. The attack on Dyn's Domain Name System (DNS) infrastructure disrupts a long list of high-profile online services, including Amazon Web Services, Twitter, Spotify, Paypal, Netflix, CNN, the New York Times, Yelp, and so on [46]. More dangerously, the source code of the Mirai botnet was made open-sourced [24], much similar to what happened to Zeus and SpyEye in the past, which sparked a series of attacks launched by the botnets in the family [9].

Botnet has the ability to maintain its virulence by evolving its structure and protocols over time. One component of a botnet that has been through many evolutions is C&C channels. A botnet C&C channel accommodates communications between bots and bot masters, which differentiate botnets from other malwares. The communication channels provide botnets the ability of updating its malicious code and protocols, allow bots to perform attacks simultaneously under the control of a botmaster. Thus C&C channel one of the targets of security researchers in order to take botnets down. Earlier botnets use Internet Relay Chat (IRC) as their C&C protocol. Eventually, as this protocol and botnet structures became obsolete and started to be detected easily, botnets abused a wide range of other protocols from HyperText Transfer Protocol (HTTP), HTTP Secure (HTTPS) to Peer-to-Peer (P2P) [47]. More recently, social network joins the list of communication channels that botmasters use for C&C [13].

In general, botnets have two main architectures, or C&C infrastructures: Centralized and Decentralized. In the centralized architecture, all bots establish their communication channel with one or a few central control servers typically over IRC and HTTP/HTTPS protocols. The obvious advantages of this topology are speedy command propagation and synchronization. However, while most earlier botnets are centralized, decentralized C&C is increasingly employed in recent years to overcome central point of failure problem. By utilizing P2P protocols to allow each node in a botnet act as a client or a master, decentralized C&C provides great flexibility and robustness. Moreover, a botnet topology can be a hybrid model of the two architectures to combine advantages of both C&C models.

Given the threats posed by botnets, botnet detection has become a critical component in network security solutions. Machine learning-based approaches are used for their ability to learn underlying patterns of data and adaptation to the dynamic nature of modern botnets. Moreover, to identify novel botnets in particular, and malicious network activities in general, anomaly detection systems based on unsupervised machine learning methods are gaining more and more interest [4].

In this thesis, a data driven approach for network traffic behaviour analysis based on an unsupervised neural network technique, namely Kohonen's Self Organizing Map (SOM) [31], is investigated. Having a unique combination of unsupervised learning and topological preserving visualization capabilities, the Self Organizing Map (SOM) is a promising technique to support network traffic analysis in a wide range of conditions and cyber-security environments.

Based on the available amount of ground truth, the thesis will examine the proposed approach in different aspects: to work as traffic behaviour classification system, or an anomaly detection system, or a clustering technique with visualization capabilities for supporting security experts in finding the threats. Moreover, the effect of different data acquisition mechanisms in identifying malicious traffic behaviours is studied by using three training schemes under unsupervised learning paradigm. Furthermore, the approach is evaluated on modern publicly available data sets, which contain not only botnet captures but also captures of other network attack behaviours, with and without packet content.

The remainder of the thesis is organized as follows. Chapter 2 summarizes the related work on malicious behaviour detection and applications of the SOM in this field. Chapter 3 discusses the methodology, whereas Chapter 4 presents the evaluation and results. Finally conclusions are drawn and the future work is discussed in Chapter 5.

Chapter 2

Related Work

Network malicious detection approaches have evolved extensively and expeditiously to cope with the advancement in malware architectures and protocols, as well as the sophistication of new attack vectors. In this chapter, Section 2.1 surveys different techniques for network malicious behaviour detection in the literature, with focus on network traffic analysis systems, at both host and network levels. Applications of the SOM and related methods in the field are summarized in Section 2.2. Finally, in Section 2.3, summary of previous approaches in the literature is presented with current limitations. From that, the novelty and functionality of this thesis in addressing open network security issues are introduced.

2.1 Network Malicious Behaviours Detection

2.1.1 Signature based approaches

Network Intrusion Detection Systems (NIDSs), from early researches to most of products nowadays, are mainly based on searching for a known set of identities, or signatures within network packets to identify malicious activities. Snort [8] and Bro [40], which are open source network-based deep packet inspection systems for intrusion detection, are two of the most notorious examples of this category. The systems depend on predetermined rule sets and policy scripts for not only intrusion detection but also forensic investigations and traffic baselining. As the tools are equipped with many rules/policies which aim to cover a wide variety of possible network conditions, the users need to determine the needs for specific features/rules and enable them accordingly to match their network conditions.

Based on Snort, Gu et al. [19] used a botnet life-cycle model to develop BotHunter. The tool correlates alerts generated using a tailored version of Snort to detect botnets and other coordination-centric malwares based on the assumption that most of the malwares follow a specific infection life-cycle model from initial inbound scan to attack behaviours. The concept is called dialog correlation, which helps to significantly reduce the number of Snort alerts, and hence generate meaningful evidences to declare a host infected. It should be noted that alongside Snort, BotHunter makes extensive use of Internet Protocol (IP) address black lists and custom tuned plugins for malicious behaviours detection.

On the other hand, Wurzinger et al. proposed a botnet detection model based on the observable command and response patterns of the botnet communications [57]. Based on the assumption that every bot receives commands from the botmaster to which it responds appropriately, the work attempts to build the malware traffic patterns by identifying hosts' responses and inspecting the preceding traffic. Apparently, deep packet inspection is required for identifying command and response tokens in the traffic that match specific signatures. The system then automatically derive signatures for detecting such command and control behaviours, and deploy them in Bro for detecting future events. On their IRC, HTTP and P2P botnet data sets, which are generated in a controlled environment, the automatically extracted detection signatures outperform BotHunter.

Although signature-based Intrusion Detection System (IDS) is very efficient detecting known attacks, which have been analyzed by security experts to release appropriate signatures/rules/policies, the approach is highly vulnerable against "zero-day" attacks. Moreover, the systems are heavily depend on receiving frequent signature updates, much like anti-virus solutions, to maintain the detection ability on even small variations in attack vectors. Thus machine learning techniques naturally found their applications in the field for the ability to automatically learn from data and extract patterns that can be used for distinguishing attack behaviours.

2.1.2 Automatic learning approaches

The same group of authors of BotHunter proposed BotMiner, an approach based on group behavior analysis, which combines both packet payload and network flow monitors for botnet detection [18]. Based on the assumption that the bots communicate with C&C servers/peers and perform malicious activities in a similar or correlated way, the work employs clustering approaches to find similar communication behaviors, as well as network activities. The model works on two planes: C-plane for clustering similar communication traffic, and A-plane - which is based on Snort - for clustering malicious activities. Cross cluster correlation is then performed to identify the hosts that share both similar communication patterns and similar malicious activity patterns. The model obtained promising results on evaluation data sets combined from campus traffic and botnet traffic generated in sandbox environments.

Strayer et al. developed a system that employ both classification and clustering techniques for detecting IRC botnet [48]. The system performs gross, simple filtering to reduce the amount of traffic flows to be analyzed before applying computationally intensive machine learning algorithms to classify the flows in "chat" group. The flows are then correlated to find clusters of flows that share similar flow characteristics and to identify the botnet controller host. They evaluated the model, together with other machine learning classifiers - Naive-Bayes, C4.5 and Bayesian Networks - on campus data to prove the concept.

Zhao et al. investigated a botnet detection system based on packet header information and time intervals [59]. Decision Tree based machine learning algorithms were utilized to generate detection models using network flow features of traffic packets. The work focused on P2P botnets on HTTP protocol, which employ fast-flux based DNS technique for their resilience. A data set consists of normal traffic from several legitimate sources and botnet traffic captures using Honeynet, [50], is the used to examine the proposed method before it is applied to live traffic detection scenarios in a web-based setting. The results shows that the system achieves high detection rates on offline data, but could also generate high false positive rate on unseen botnet traffic.

Also based on Netflow to detect P2P botnets, Nagaraja et al. proposed BotGrep - a monitoring framework that construct graphs of traffic patterns [38]. The work takes advantage of a key feature that modern botnets increasingly use structured overlay topologies to localize botnet members with unique communication patterns from the built graphs. The authors applied the model on Internet service provider sized benign network traffic, with Honeynet botnet traffic injected to demonstrate the concept. Moreover, a privacy-preserving extension was also introduced to simplify deployment across networks. It is noteworthy that the model based on computationally extensive graph algorithms, which can take days to build and manipulate the graphs before the conclusions can be drawn.

Haddadi et al. employed two machine learning algorithms, namely C4.5 Decision tree, and Symbiotic Bid-based Genetic programming, for building detection models [21]. The objectives are not only to apply the genetic programming to improve the state-of-the-art in intrusion detection, but also to find the feature sets that best describe the botnets and to return a solution that is suitable for a signature-based botnet detection system. Furthermore, in [20], the authors confirmed the advancement of proposed method over packet payload inspection based systems.

Recently, in [58], Yan et al. proposed PeerClean - a multi-phase detection system targeting peer-to-peer botnets. The first phase clusters individual connections or hosts with similar flow traffic statistics into groups. Then PeerClean extracts collective connection patterns from each group using a proposed dynamic group behaviour analysis method. Finally, a Support Vector Machine (SVM) classifier is trained using group connection patterns to identify botnet groups. The system is shown to be effective in detecting several known botnets in a mixed data which consists of traffic captured from an edge router of a campus network with known P2P applications (benign), and botnet data from running botnet sample in a controlled environment.

AI^2, a concept proposed in [53] by Veeramachaneni et al. incorporates analyst's responses in an active learning detection solution. From the raw web and firewall logs, the system performs aggregation efficiently to produce features representing user behaviours. An ensemble of outlier detection methods - Principle Component Analysis (PCA)-based, autoencoder-based, or density-based outlier analysis - is used to discover suspicious instances and then present them to security analysts to obtain feedback. Finally, a supervised learning module is applied to classify future instances based on the feedback and outlier indicators. It should be noted that the system requires a big data infrastructure to accommodate the resource extensive learning algorithms and a moderate amount of expert knowledge and learning time to improves the detection performance.

While most of the machine learning approaches for botnet detection are based on supervised learning, unsupervised learning approaches have also found their applications in the field, especially in anomaly detection systems. Leung et al. proposed a density-based and grid-based clustering algorithm to discover the characteristics of the majority of connections in network traffic [34]. A clustering algorithm, fpMAFIA - a modified adaptive grid algorithm similar to Apriori algorithm [1], is used to mine frequent pattern tree of connection records. The characteristics of malicious clusters are used to classify future connections. Evaluated using the 1999 KDD Cup data set, the technique produced comparable results to existing supervised approaches but also suffered from a high false positive rate.

Perdisci et al. proposed using an ensemble of one-class SVM for payload-based anomaly detection systems [42]. They used a clustering algorithm originally proposed for text classification problems to reduce the dimensionality of the feature space obtained from n-gram analysis of payload. Then the anomaly detectors are applied to each description of the payload to produce aggregated results. It has been shown that the system improved both the detection accuracy and the hardness of evasion of the payload-based anomaly detection system.

2.2 Applications of SOM in network security

Kayacik et al. proposed an approach to network intrusion detection based on a hierarchy of SOMs [30]. Using 1999 KDD Cup data set for training, two hierarchical SOM architectures were proposed. The first architecture uses only six basic features from the data set and generates a three-layered SOM hierarchy, where the first layer SOMs are used to generalize data from each feature individually. Output of each first-level SOM is clustered to six clusters for higher layer training. The second architecture uses all 41 features to directly train a two-layer SOM model, which is similar to the second and third layers in the first model. The proposed method achieved the best performance to date of a detector based on an unsupervised learning algorithm on the KDD data set.

Based on the Growing SOM - a variant of the SOM - Ippolity et al. developed a threshold based training approach, namely Adaptive Growing Hierarchical SOM, for building an online network intrusion detection system [27]. In their work, system parameters are adjusted dynamically by using quantization error feedback to adapt to the new training data. A dynamic input normalization process is applied to accommodate live training conditions. Furthermore, the SOM units are monitored using the proposed confidence filtering and forwarding mechanism. The results on 1999 KDD Cup data set and their own simulated traffic in a testbed network show enhancement over performance of previous approaches.

Recently, a combined approach of PCA and probabilistic SOM for network intrusion detection is introduced by De la Hoz et al. in [10]. In the model, feature selection and noise removal are carried out using PCA and Fisher discriminant ratio. The SOM is then employed to model the feature space and represent different types of connections, including both normal and anomalous connections. From the SOM, a Gaussian mixture model is calculated for classification, which allows modifying detection priorities by adjusting the units posterior activation probabilities. On NSL-KDD data set, which is a modified version of 1999 KDD Cup data set with balance training set, the experiment results shows a detection accuracy of up to 88%.

2.3 Summary

While machine learning has been studied broadly in network security, there exists several issues in the literature that inhibit a widespread deployment of such systems:

- Many approaches are heavily dependent on deep packet inspection [10][18][34][42][57], which makes them become nearly impossible to be applied to modern network environments, where almost all traffic from both normal users and malwares is encrypted. Hence, an approach not utilizing encrypted payload information, e.g. based on network flows, may improve the state-of-the-art in traffic analysis.

- Some proposed systems, [38][48][58][59], focus only on a specific botnet architecture/ botnet connection protocol. One notable instance of outdated communication protocol that is not employed by modern botnet is IRC.

- Many previous works were built upon and tested against outdated data sets, especially the 1999 KDD cup data set [34][30][27][10], or data sets from closed environments [19][18][57][48][42][58]. It should be noted that the 1999 KDD cup data set has many drawbacks that have been pointed out in [37]. Moreover, the data set is already well investigated by proposed detection models using packet statistics. This raises the question about performance of such systems on new publicly available data sets representing modern botnets.

- Many proposed systems employ sophisticated and computational extensive analysis architecture [53][18][38][58][27][48]. The models are also based on specific assumptions about correlation and synchronization in traffic patterns. Those reasons limit the models from becoming a ubiquitous network traffic analysis and classification system.

Furthermore, the fact that in practice either there is no labelled data or there is very few of them reduces the versatility of systems employing supervised learning techniques. On the other hand, approaches based on unsupervised learning in the aforementioned works provided comparable results to that of supervised learning approaches, while enable an intrusion detection system to potentially generalize the learned data for recognizing novel threats, i.e. anomaly detection.

Applying SOM - an unsupervised learning method with strong visualization capabilities preserving the key topological relationships of the data [31], potentially would be one of the most supportive features for a human expert to analyze the data. Therefore, the novelty and the functionality of the approach in this thesis lie in the following aspects:

- The thesis attempts to explore further in unsupervised learning and visualization landscape, thus concurrently seeking a simple solution that is less dependent on expert knowledge and labelled data, and also more flexible in deployment.

- The thesis relies on only network traffic flows, which is statistics exported from network packet headers solely but not the packet content, hence possess the ability to overcome the curse of encryption.

- The thesis explores a wide range of data acquisition conditions that can happen in real-world situations. The approach in the thesis is examined against diverse data sets of not only modern botnets but also web attack behaviours.

- The capability of the approach in supporting security experts in analyzing unknown/unlabelled data is also explored, by applying the model for recognizing unseen botnets, and investigating unlabelled majority portions of the employed data sets. The process also attempts to show the capacity of topographic preserving visualization ability of the SOM for network security.

Chapter 3

Methodology

The principle interest of this research is to explore the capabilities of an unsupervised learning approach as a data analytics tool for network and system behaviour detection using minimal *a priori* information. The data-driven approach based on unsupervised learning enables network administrators to discover threats and malicious behaviours in circumstances where experts' knowledge is scarce or nonexistent. To this end, Self Organizing Map (SOM) is employed to build an analysis system, not only supported by its unsupervised nature but also its abilities in visualizing the data. Different SOM training schemes are used for evaluating real-world data acquisition strategies for behaviour detection and to overcome the lack of well-labelled data. Moreover, the work is focused on network traffic flows instead of raw network data (packet payload) for increasing capabilities against network traffic encryption. In the following, the employed data sets and network traffic flow will be presented in Section 3.1. The learning algorithm is described in Section 3.2. The proposed system architecture is presented in Section 3.3.

3.1 Network Traffic Flow and Data Sets

3.1.1 Network traffic flows

Two network traffic data sets in this research (ISOT and CTU13) are exported as network traffic flows. A flow is defined as a logical equivalent to a call or connection, which connects a pair of terminals and contains a group of features [5]. Flows are commonly identified by a set of five different attributes (5-tuples) including source and destination IP addresses, source and destination Transmission Control Protocol (TCP) / User Datagram Protocol (UDP) port numbers, and the protocol, over a predetermined duration. Flow features typically include descriptive statistics that are calculated from aggregating Network and Transport layers header information of the packets in a flow.

The use of low layer packet header properties (especially Network and Transport layers) as descriptive characteristics of a flow makes the approaches based on flow Application layer independent. Moreover, given that network traffic encryption is very popular in both benign applications, for protecting users' privacy and sensitive information, and malicious ones, for hiding from the detection systems that analyze network packet payload, the detection approach using only flow exported from packet headers may improve the state-of-the-art in unsupervised learning based network and system analysis. One can imagine network flow as the summary of network connections between hosts in a form that is not directly decided by the packet content. So, although botnet master can effectively change a part of malicious packet payload, or encrypt the connection to evade signature based detection systems, it would be much harder to change the abstract depiction of network connections formed by the flows. The reason is that ultimately the malicious connections are still established automatically by predefined code, and hence would be different from diverse normal user and computer behaviours.

Flows can be extracted from network traffic using proprietary flow exporters in routers and switches, e.g. Cisco Netflow [7] and Juniper JFlow[28]. On the other hand, open source flow exporters can also extract flows from network traffic captured at network devices as well as network terminals with a wide variety of features and great flexibility. Good performances achieved by such flow exporters, e.g. Argus [43] and Tranalyzer [6], were confirmed by Haddadi et al. in [22]. Hence in this thesis, Tranalyzer is employed for exporting the flows from raw network traffic. The lists of features exported using Tranalyzer, as well as the features employed in this research are presented in Appendix A.1.

3.1.2 Data sets

Obtaining high quality data for designing and evaluating attack behaviour detection systems typically involves considerable difficulties. One reason is that companies and organizations are prevented from sharing network traffic by agreements protecting users' identities and privacy. Moreover, even when data is published / shared, it may come with no additional information about the contained activities. Hence, it is still hard to identify the correct labels (ground-truth) for training and testing purposes.

In this thesis, three publicly available network traffic traces are used. First two sets, CTU13 and ISOT contain botnet traffic, while the remaining data set, HTTP-CSIC contains web attack traffic. The data sets ensure a wide range of malicious as well as normal behaviours and categories, both with and without traffic payload. This enables the proposed SOM based approach to be evaluated under different network security scenarios and applications.

CTU13

The CTU13 botnet traffic data sets were captured in 2011 by Malware Capture Facility Project of Czech Technical University [15]. The goal was to have a large database of real botnet traffic mixed with normal traffic and background (unidentified) traffic. These data sets, which are referred as CTU13*a-m* in this thesis, consist of thirteen traffic traces of different botnet samples. Under each scenario (botnet sample), a specific malware was executed in a set of virtualized Microsoft Windows XP computers in a Linux Debian host. Traffic from each virtual machine, which is exclusively malicious, was being bridged into the university network. The traffic was then captured both on the Linux host and on one of the university routers. The final data set is the one captured from the router, with botnet label information taken from traffic of the Linux host.

The main characteristics of the scenarios and their behaviours are shown in Table 3.1. Due to privacy limitations, only the network flow files containing basic flow features extracted using Argus are published by the CTU. The features are: *the duration, port numbers, the direction, source and destination types of services, the number of packets, the number of bytes, the number of source bytes, and the protocol.* The list of features is supported by almost every traffic flow exporters. Thus, the capability of a model employed such features is not limited to Argus but any flow exporter that is applicable to specific network deployment.

All of the provided features, except the direction, are employed in this thesis. By using only the provided basic flow characteristics, the intention is to test the performance of the proposed approach using minimum *a priori* information. By minimizing the *a priori* information, the thesis aims to minimize the blind sights and not to miss the new (unknown) malicious behaviours.

The labelling process of the CTU13 data set, as discussed in García et al. [15], is summarized below:

1. Assign the Background label to the whole traffic.

2. Assign the Normal label to the traffic that can certainly be identified as traffic from the known and controlled computers in the network, such as routers, proxies, switches, and their personal computers in their laboratory.

3. Assign the Botnet label to all the traffic that comes from or to any of the known infected IP addresses.

Hence, in CTU13 data set, there is a large portion of data for further exploration, because ground-truth is not known for this portion. Garcia et al. labelled this portion as Background. In other words, the Background traffic flows of these data sets may contain either normal or malicious behaviours. This portion (background) is also referred as the unknown portion of the data in this thesis.

ISOT botnet data set

ISOT botnet data set, provided by University of Victoria [59], is the combination of several publicly available malicious and non-malicious data sets, including data sets from the Traffic Lab at Ericsson Research in Hungary[49] and Lawrence Berkeley National Laboratory (LBNL) [33] for legitimate and background traffic, and Storm and Waledac botnet traffic from the French chapter of honeynet project [50]. Waledac and Storm were two of the most prevalent P2P botnets with decentralized communication protocols. While Storm using the old-fashioned peer-to-peer Overnet as its communication channel, its successor Waledac utilizes HTTP and a fast-flux based DNS network for concealing malicious activities. The Ericsson Lab traffic contains general traffic from a variety of applications, including HTTP web browsing behaviour, World of Warcraft gaming, and popular bittorrent clients such as Azureus. Additional non-malicious background traffic is also incorporated from the LBNL trace data, which contains network traces for a variety of activities spanning from web and email to backup applications as well as streaming media in an enterprise network environment. The merging process of ISOT data set from the component sets is as follow:

Table 3.1: CTU13 data sets description ([15]). CF: ClickFraud, PS: Port Scan, FF: FastFlux, ICMP: Internet Control Message Protocol.

Scen.	Bot	IRC	SPAM	CF	PS	DDoS	P2P	HTTP	Note
a	Neris	✓	✓	✓					
b	Neris	✓	✓	✓					
c	Rbot	✓			✓				
d	Rbot	✓				✓			UDP and ICMP DDoS. Scan
e	Virut		✓		✓			✓	Scan web proxies.
f	Menti				✓				Proprietary C&C. RDP.
g	Sogou							✓	Chinese hosts.
h	Murlo				✓				Proprietary C&C. Net-BIOS, STUN.
i	Neris	✓	✓	✓	✓				
j	Rbot	✓				✓			UDP DDoS
k	Rbot	✓				✓			ICMP DDoS.
l	NSIS.ay						✓		Synchronization.
m	Virut		✓		✓			✓	Captcha. Web mail.

first, the IP addresses of the infected machines in the botnet data are mapped to two of the machines providing the background traffic, second, all of the network trace files are replayed using the TcpReplay on the same network interface card in order to homogenize the network behaviour in the component sets. This replayed data is then captured again using Wireshark [59].

HTTP-CSIC

The HTTP data set CSIC 2010 from the Information Security Institute of Spanish Research National Council (CSIC) [16] contains thousands of web requests for testing web attack protection systems. The normal and anomalous web requests are automatically generated for a set of web page on a server from databases that contain parameters extracted from actual normal and anomalous web requests. The web attacks were generated based on web security assessment tools such as Paros and W3AF. The attacks are targeted to an e-Commerce web application developed at CSIC. The data set is generated automatically and contains 36,000 normal requests and more than 25,000 anomalous (attack) requests. This data set includes attacks such as Structured Query Language (SQL) injection, buffer overflow, information gathering, file disclosures, Carriage Return Line Feed (CRLF) injection, Cross Site Scripting (XSS) attack, server side include, parameter tampering and so on. There are three types of anomalous requests included in the data set:

1. Static resource requests, which try to request hidden (or non-existent) resources. These requests include obsolete files, session identifier in Uniform Resource Locator (URL) rewrite, configuration files, default files, etc.

2. Dynamic resource requests (attacks), which modify valid request arguments: SQL injection, CRLF injection, XSS attack, buffer overflows, etc.

3. Unintentional illegal requests. These requests do not have malicious intention, however they do not follow the normal behaviour of the web application and do not have the same structure as normal parameter values (for example, a telephone number composed of letters).

From the provided requests in the data set, numerical representation vectors are extracted using a set of heuristically determined features for training and testing the proposed system. The features can be found in Appendix A.2.

3.2 Learning Algorithm - Self Organizing Map

Self Organizing Map (SOM), or Kohonen's map is one of the most popular unsupervised neural network models [31]. The algorithm is based on unsupervised, competitive learning to produce a non linear, ordered, low dimensional (typically two-dimensional) similarity map projection of multi-dimensional input space, which can also be called the output space. The SOM output space consists of nodes or neurons which can act as decoders or detectors of their respective input space domains after the training process. Hence, the SOM provides visualization and summarization options for high dimensional data with topological relationships preserved.

Figure 3.1 presents a schematic representation of a self organized map. Each node in the SOM output space has a weight vector with the same number of dimensions as the input vector, as well as a fixed position in the map plane. The SOM algorithm is performed in an iterative basis in order to form the final map. In an iteration, for each training vector, the algorithm calculates distances between input vectors and SOM nodes to choose the Best Matching Unit (BMU). The algorithm then updates the weight vectors of the BMU and its neighbours accordingly for the training process.

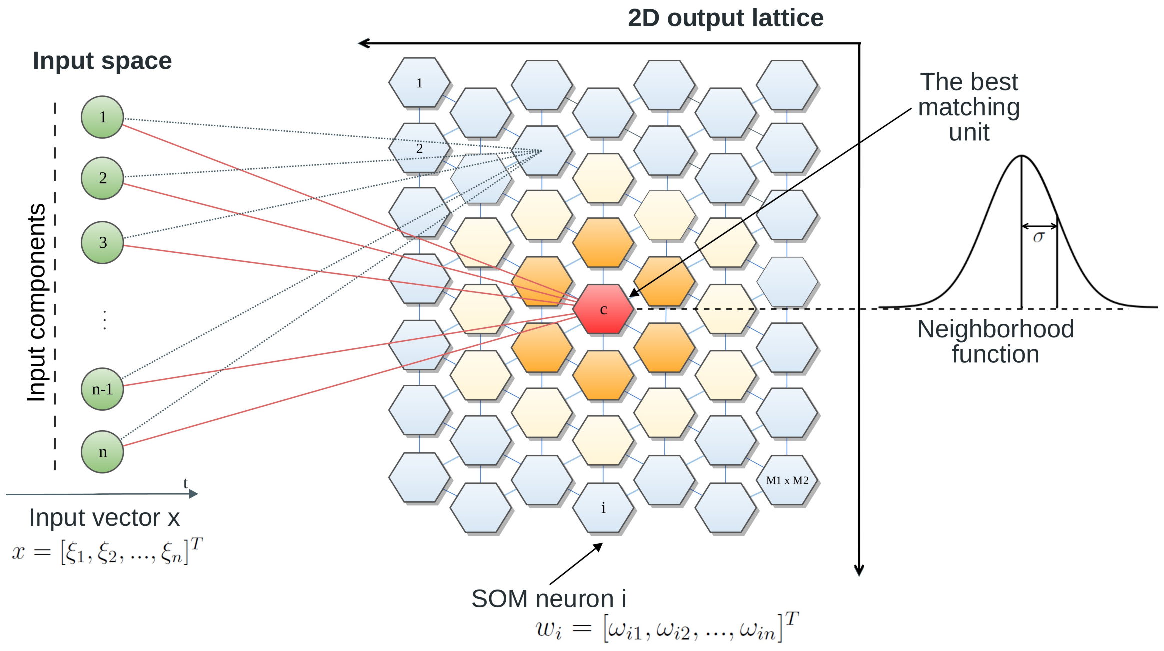


Figure 3.1: A schematic representation of a self-organizing map

Define the training set $\mathcal{X}$ as a list of all input vectors $x = [\xi_1, \xi_2, ..., \xi_n]^T \in \mathbb{R}^n$. The original SOM iterative learning procedure can be summarized as follows:

Step (1) Initialize a $M_1 \times M_2$ two-dimensional, typically a hexagonal or rectangular, lattice $\mathcal{W}$ of neurons, each has a weight vector $w_i = [\omega_{i1}, \omega_{i2}, ..., \omega_{in}]^T \in \mathbb{R}^n$ and a position r_i in the 2D plane. The weight vectors can be assigned randomly or linearly based on input range.

Step (2) A random input vector x is presented to train the lattice. A distance measure is calculated between x and all the SOM nodes. One popular choice for the distance measure is the Euclidean distance, $d(x, w_i) = ||x - w_i||$, where $||x|| = \sqrt{\sum_{j=1}^{n} \xi_j^2}$. Then the winning node w_c, or the BMU, is identified by minimum distance to the input vector.

$$c = \arg\min_i d(x, w_i). \tag{3.1}$$

Step (3) Weight vectors of the winning neuron and its neighbours are adjusted according to the input vector:

$$w_i(t+1) = w_i(t) + h_{ci}(t)(x - w_i(t)), \tag{3.2}$$

where hc_{ci} is the *neighbourhood function* and $t = 0, 1, 2, ...$ is the discrete time coordinate. Acting as a smoothing kernel defined over the lattice points, the neighbourhood function plays a vital role in the SOM convergence process. It defines the magnitude of influence each input training vector has over the SOM nodes, and implicitly the update region around the BMU. The neighbourhood function can simply include the *neighbourhood set* of neurons around node c. However, more frequently the neighbourhood function has a Gaussian form:

$$h_{ci}(t) = \alpha(t).exp\left(-\frac{||r_i - r_c||^2}{2\sigma(t)^2}\right), \tag{3.3}$$

where α_t is the learning rate factor and $\sigma(t)$ is the kernel width, also called the *neighbourhood radius*. Both $\alpha(t)$ and $\sigma(t)$ are scalar-valued and monotonically deceasing over time.

Step (4) Repeat steps (2 - 3) by a predetermined number of iterations or until the convergence criterion is satisfied, i.e. the corrections to the SOM weight vectors become zero.

The original iterative SOM algorithm is also called on-line SOM [31], as it can be applied in an on-line fashion by incrementally updating the SOM each time a new training pattern appears.

3.2.1 Batch SOM training

In practice, when the whole training set is presented at the beginning, the SOM batch training algorithm is generally preferred for faster convergence rate, less computational cost, and less number of learning parameters. Additionally, the SOM training process usually consists of two phases: coarse training, during which the high-level topographic order of the SOM is quickly formed, and fine training, for obtaining a more accurate final state. Different from the original SOM learning algorithm, steps (2-3) are performed for all data points in the training set at one:

Step (2'-3') For each SOM node i, collect a list of all training data points x whose BMU is w_i. The weight vectors of SOM nodes are then updated as follows:

$$w_i(t+1) = \frac{\sum_{j=1}^{N} h_{ci}(t)x_j}{\sum_{j=1}^{N} h_{ci}(t)}, \tag{3.4}$$

where N is the training set size.

Step (4') Repeat step (2'-3') in two phase: coarse training phase with large neighbourhood radius $\sigma_{coarse}(t)$ and small number of iterations l_{coarse}, followed by fine training phase with small and constant neighbourhood radius $\sigma_{fine}(t)$. The fine training phase can have higher number of iterations l_{fine}, or is continued until convergence.

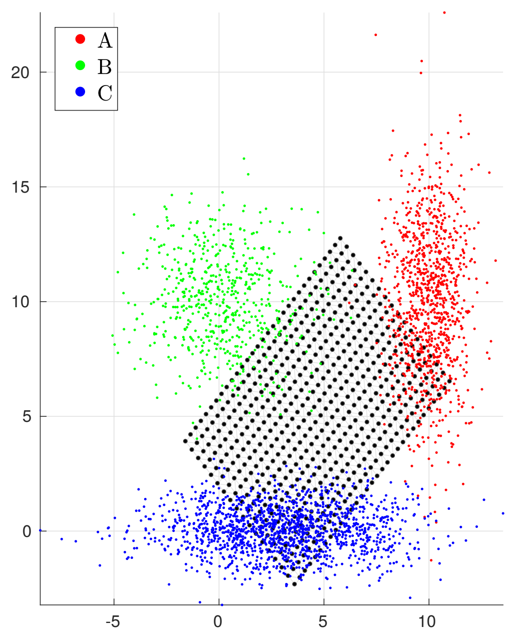
The batch SOM algorithm is employed for this work. The parameters and the initialization method are presented in Section 4.1

3.2.2 Characteristics and visualization capabilities of SOM

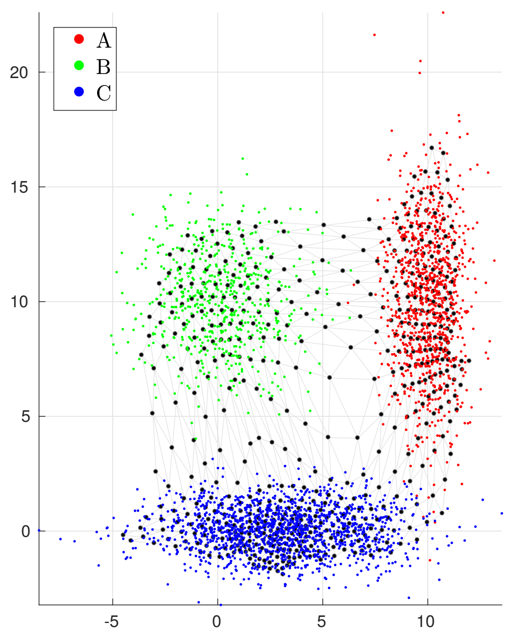
Post training, SOM preserves the topological properties of the input space, and therefore can be used as a data analytics tool to visualize and analyze the high-dimensional data. Moreover, SOM has the ability to generalize data from the training set. Characteristics of each new input can be derived by identifying its BMU and quantization error. Quantization error of each data instance is defined as the distance to its BMU, hence it quantifies the similarity between the data instance and the SOM [31].

In this section, a synthesized data is used to demonstrate the visualization capabilities of SOM. For simplicity and clarity, it is assumed that the input data has two dimensions and is distributed in three different clusters as in Figure 3.2. Using linear initialization, the initialized SOM lattice is shown in Figure 3.2(a). Post training, the SOM lattice is adjusted to represent the data, as shown in Figure 3.2(b).

The distance matrix is calculated between weight vectors of SOM nodes to reveal the structure of the trained SOM post training. Longer distances indicate less similarity between the weight vectors. Basically, for a SOM of size $M_1 \times M_2$, the distance matrix has size $(2M_1 - 1) \times (2M_2 - 1)$. The SOM distance matrix for the SOM trained



(a) Linearly initialized SOM lattice



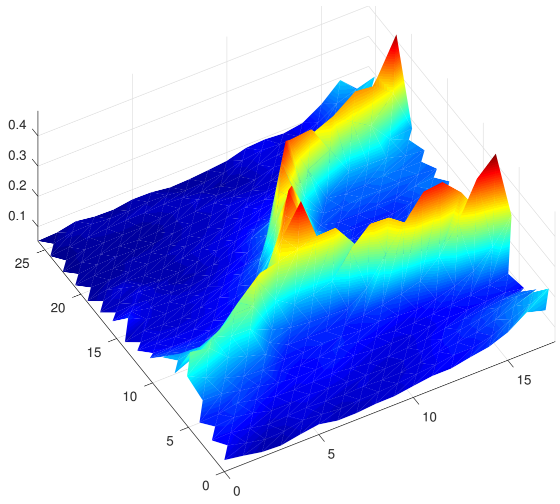
(b) SOM lattice post training

Figure 3.2: The SOM lattice and the synthetic data

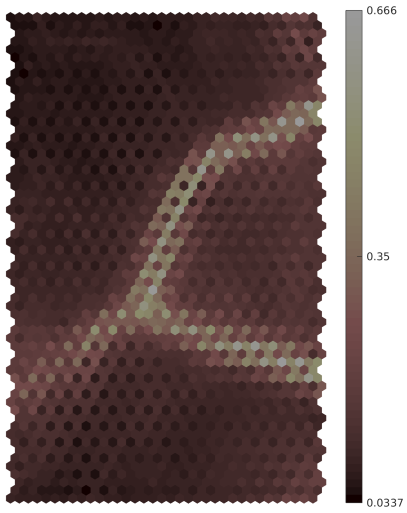
on the synthetic data is shown in Figure 3.3(a). In practice, the distance matrix usually visualized as U-matrix [51]. Each node in the U-matrix represents one element in the distance matrix, Figure 3.3(b), where the colour bar on the right shows the distance range. The lighter the colour gets, the longer the distance becomes. In this example, it is clear that the U-matrix successfully illustrates the degree of clustering tendency on the trained SOM, which resembles three clusters in the data.

The SOM can also be visualized based on the distribution of data points on SOM nodes, as shown in Figure 3.4(a). In this case, the size of each SOM node w_i is determined by the number of data points whose BMU is w_i. In this thesis, *hit map* (Figure 3.4(b)), a combination of the U-matrix and the SOM hit distribution, is extensively employed for visualizing the SOMs. On a hit map, the background colour represents an interpolated shading version of U-matrix, while the size and colour of each node represent volume and class label of best matching data for each SOM node, respectively.

The example showed that the SOM is an effective unsupervised learning method for data visualization and exploration. Not only preserving the topographic relationships in the data, the SOM also provides the ability to detect "*anomalous*" data

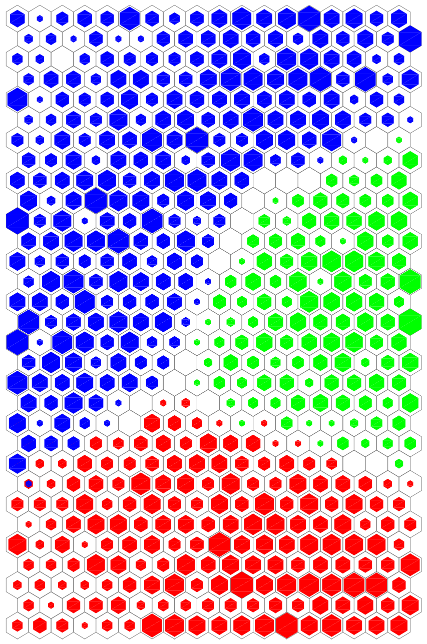


(a) 3D representation of SOM distance matrix

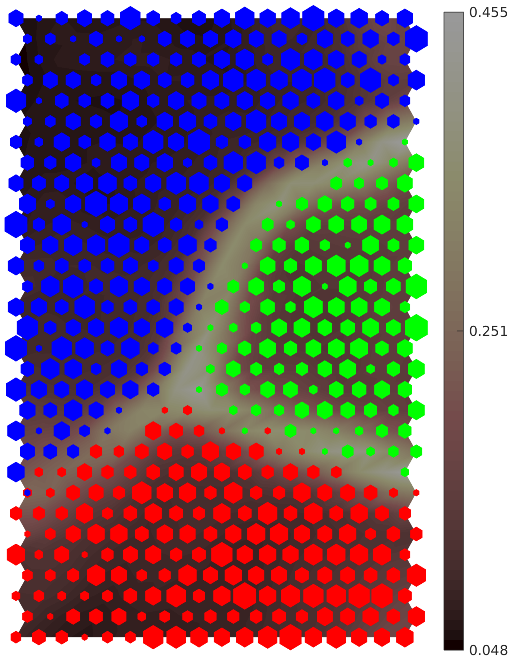


(b) SOM U-matrix

Figure 3.3: SOM Distance matrix (U-matrix)



(a) Data distribution on the trained SOM



(b) Hit map

Figure 3.4: Visualization of data distribution on the trained SOM

points, which are different from the training data by having high quantization errors or by matching the lighter regions on the SOM U-matrix.

3.3 System Architecture

The proposed system architecture is shown in Figure 3.5. Fundamentally the system is based on a *data-driven* approach using unsupervised learning with visualization abilities (SOM) to learn network and system behaviours with minimum expert knowledge.

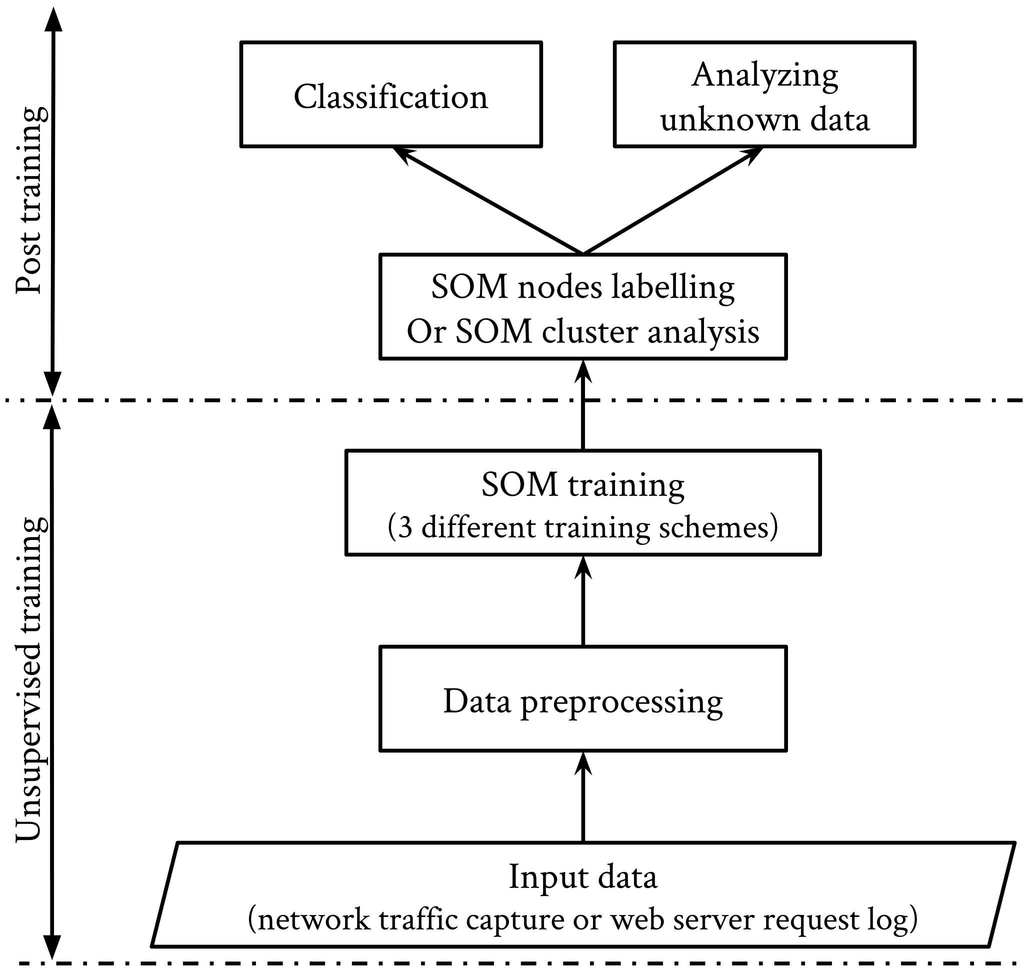

Post training
Unsupervised training
Classification
Analyzing unknown data
SOM nodes labelling Or SOM cluster analysis
SOM training (3 different training schemes)
Data preprocessing
Input data (network traffic capture or web server request log)

Figure 3.5: Proposed system architecture

Adhering to the unsupervised paradigm, the system architecture is kept minimal and straightforward. Raw input data is processed to numerical vectors, which are network flows from traffic captures or request characteristics for web log files. The data is then pre-processed, e.g. for normalization purposes or for dimensionality reduction purposes (detailed later in the thesis), before being input to the SOM learning algorithm. The original data distributions are always kept intact. Hence the SOM is obtained post training in a completely unsupervised manner. Since no label information or human-defined knowledge of data is used for the proposed SOM training process, only one layer of SOM with sufficient resolution is employed. This is supported by the presumption that learning from data, SOM may form well-separated node regions to differentiate distinct communication behaviours.

To quantify the system performance, if the ground-truth of the training data is available, it can be applied for labelling the SOM nodes post training. The labelled SOM can then be used to classify unseen testing sets. Details of the SOM nodes labelling process will be presented in Section 4.1.2.

On the other hand, when label information is not available, cluster analysis based on the trained SOM topology visualization, e.g U-matrix (Figure 3.3(b)), can be used along with expert knowledge and information from other sources to derive meaningful insights from the data. Moreover, possessing powerful visualizing capabilities, the trained SOM can be employed to investigate the unknown data, using knowledge learned in the training and labelling phases.

3.3.1 Training schemes

Hackers are employing more and more sophisticated techniques to hide malevolent software and any fingerprints (evidence) that might be left by the attack performed [47]. This results in the malicious traffic becoming more and more similar to legitimate (normal) traffic, making the identification established in previous works blurry. So, to shed light into this phenomena and to analyze it further, in this thesis the SOMs are trained using three different schemes based on the chosen training data:

(i) use both known normal / legitimate and known malicious traffic for training purposes, as done in the previous supervised learning approaches [30][26];

(ii) use only normal / legitimate traffic for training purposes as done in the previous unsupervised learning (anomaly detection) approaches [34];

(iii) use only malicious (botnet / C&C) traffic or anomalous requests for training purposes as done in some of the previous one-class classifier approaches [55][42].

The reason behind these training schemes is to not only represent the real-life security conditions, but also to shed light into understanding the performance gains / losses under different types / amounts of labelling information, i.e. ground-truth. For example, the data collected by honeypots is usually only considered as being representative of malicious behaviour. On the other hand, in idealistic cases of networks where there are no attacks, the data collected contains only legitimate traffic. Moreover, even when a threat is discovered in the collected traffic, it is very challenging to fully identify the extent of the threat and label the data collected.

In summary, three different types of SOMs (trained based on the aforementioned training schemes) are employed for exploring the unknown / unlabelled traffic present in the aforementioned data sets. By analyzing the distribution of such traffic on the trained SOMs, the intention is to investigate the ability of the different training schemes on inspecting / analyzing unknown traffic for different attack and normal (legitimate) behaviours. This is the basic step toward an unsupervised system for automatically detecting anomalous behaviours in everyday traffic/ system logs. It should be noted here that the aim is to help the human expert to analyze the unknown data, but not to automatically classify. In the proposed system, the final decision (classification) is left to the human expert.

Chapter 4

Evaluation

In this chapter, first the details of SOM training and labelling, as well as data sets and performance measurements are presented in Section 4.1 and 4.2. Then, the classification results of the three mentioned training schemes on the data sets are shown in Section 4.3 before the performances of the SOM on the data sets are compared to other well known supervised and unsupervised learning algorithms in Section 4.4. Next, Section 4.5 presents the performance evaluations of the proposed system and a hierarchical method which employ the SOM in a semi-supervised manner. Finally, the SOMs are used for unknown data analysis in Section 4.6.

4.1 SOM Training

The proposed system in this thesis is built based on SOM Toolbox 2.1 [54], which is developed and recommended by the developers of SOM [32].

4.1.1 SOM initialization and training parameters

The SOM training parameters used in this thesis are summarized in Table 4.1. The initialization of the SOM can be done by assigning a random weight vector, or a random input data point to each node. However, as stated in [32], SOM learning is generally faster (in terms of convergence) if regular initial values are given to the maps. Thus, the SOMs in this thesis are initialized linearly.

In SOM linear initialization method, the beginning weight vectors are uniformly distributed in a rectangle on a hyperplane made of the two greatest eigenvectors of the input data. The ratio between the sidelengths of the rectangle is decided according to the two eigenvalues, while the center of the initialized SOM is also the center of the input data, Figure 3.2. The number of the map units in this thesis is determined based on the size of the input data N using the following formula: $M_1 \cdot M_2 = 10\sqrt{N}$ [54]. Given the number of nodes, one can easily see from Section 3.2 that one batch

Table 4.1: SOM training parameters

Parameter	**Value**
SOM map	2D hexagonal lattice
Initialization method	Linear initialization
Number of nodes ($M_1 \cdot M_2$)	$10\sqrt{N}$
Neighbourhood function	Gaussian
Coarse training length l_{coarse}	100
Fine training length l_{fine}	minimum 400, the final number is decided by convergence criteria
Coarse training neighbourhood radius σ_{coarse}	$0.25 \cdot \max(M_1, M_2)$ to $\max(1, 0.05 \cdot \min(M_1, M_2))$
Fine training neighbourhood radius σ_{fine}	$\max(1, 0.05 \cdot \min(M_1, M_2))$

SOM training iteration would take $\mathcal{O}(N\sqrt{N})$ time. Hence the total SOM training time is $\mathcal{O}((l_{coarse} + l_{fine})N\sqrt{N})$.

4.1.2 Labelling the SOM nodes

As described in Section 3.3, three different training schemes based on three distinct training data compositions are employed. For each training scheme, when the ground-truth of the input data is available, post training, the SOMs can be labelled as the following:

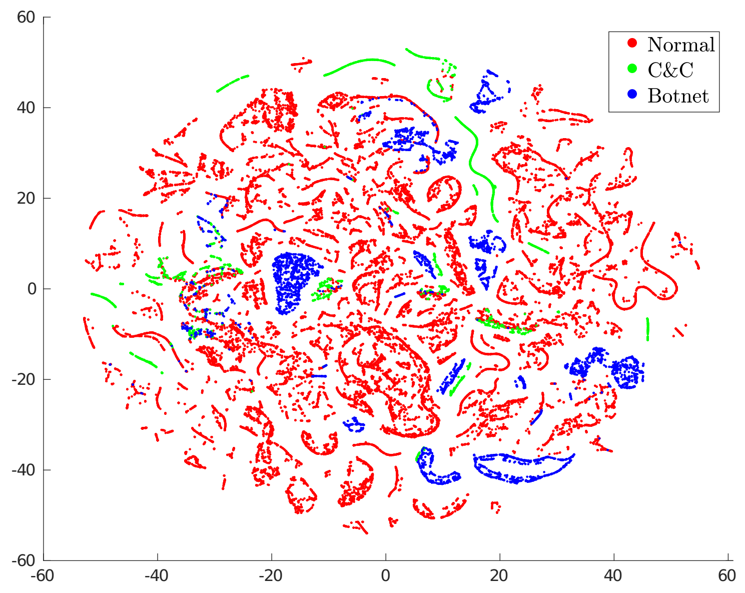
- **Training scheme (i)**, using both Normal and Malicious data: post training, for each SOM node i, the set of labelled training examples whose BMU is w_i is collected. From this set, votes toward different classes are calculated with training data distribution taken into account to offset the skewness in the data. For example: If node i is the BMU of 50 examples, out of which 18 are labelled “A” and 32 are labelled “B”, and given that the input data has a distribution of 60% “A” and 40% “B”, then the votes are calculated as $v_{iA} = 18/0.6 = 30$, and $v_{iB} = 32/0.4 = 80$. This leads to the node being labelled as “B”. It is also noteworthy that the trained SOM usually consists of well-separated regions of different classes (Figure 3.4(b)). Thus, the votes are routinely dominated by only one class. For the remaining SOM nodes on the trained map that do not have best matching data examples, the labelling process is done based on the nearest neighbours basis. Specifically, the nodes are assigned the labels of the majority of their neighbouring nodes.

- **Training schemes (ii) and (iii)**, using only Normal or Malicious data, respectively: Since the training data of these two training schemes contains only one class, the SOMs are labelled by adopting an outlier detection principal. Specifically, a threshold τ is determined for each trained SOM to specify the set of important map units, which represent the core behaviours of the training class. Basically the set of important map units is defined as the minimal set of SOM nodes with the greatest hits that are BMUs for at least $\tau \cdot N$ training data points. Although the value of τ varies from one data to another, the principle assumed here is that τ needs to be greater or equal to 0.9 to retain a sufficient number of nodes representing the behaviours of the training class. The selected τ values will be justified by the Receiver Operating Characteristic (ROC) curves. Naturally, any number of mechanisms could be assumed for outlier detection [36], the thresholding approach adopted here represents a convenient starting point. Future work could conduct a wider study of the relative significance of assuming different approaches.

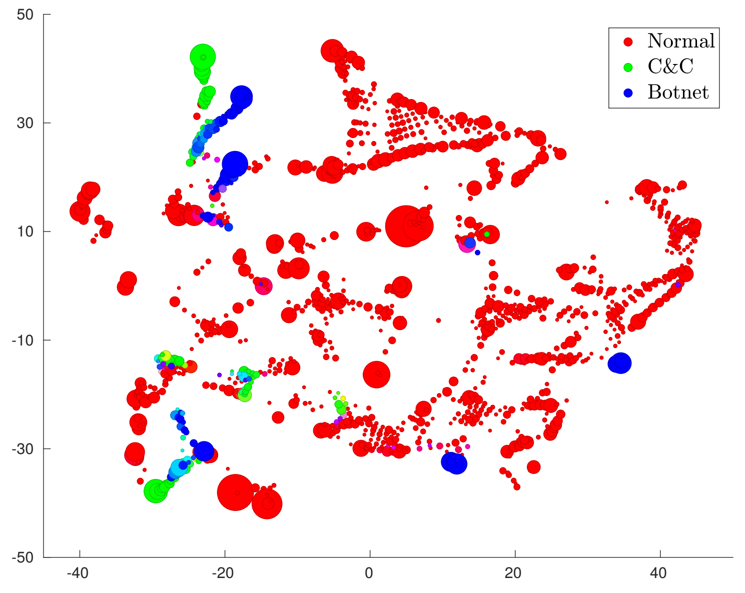
4.1.3 Verification of SOM's learning ability

To verify the performance of the SOMs post training, tSNE [52], an independent non-parametric mapping method for data visualizing, is employed. Although being limited by non-parametric nature, which prohibit the mapping from applying to unseen data, and expensive runtime, tSNE is prominent for its ability to produce 2D mappings that captures the structure of the high-dimensional input data. Figure 4.1, 4.2, 4.3 presents the tSNE mappings of the original data sets, as well as the mappings of the codebooks[1] of the SOMs trained on the respective data sets. Specifically, Figures 4.1(a), 4.2(a), 4.3(a) show the tSNE mappings of ISOT, CTU13a, and HTTP-CSIC respectively, while Figures 4.1(b), 4.2(b), 4.3(b) show the tSNE mappings of codebooks of the trained SOM using the corresponding data. It is noteworthy to mention that SOM and especially tSNE are capable of representing non linear properties. So, one shall only expect to see the structure similarity (but not the linear similarity) between the mappings of the same data set, e.g. between Figures 4.2(a) and 4.2(b). The similarity

[1]The codebook is the set of all weight vectors of the SOM.

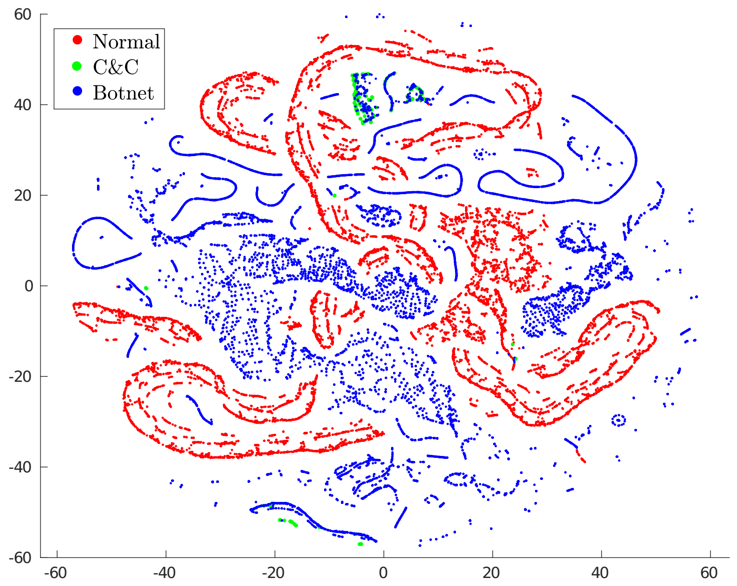


(a) tSNE mapping of ISOT

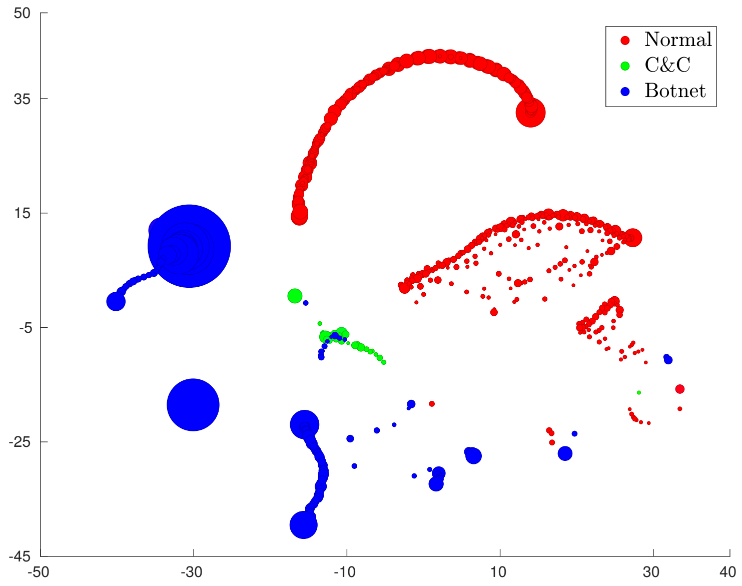


(b) tSNE mapping of the SOM trained on ISOT. The size of each node denotes the number of data points that hit the node, while the mixed colour in a node denotes that the node is the BMU for data from more than one class.

Figure 4.1: tSNE mappings of ISOT data set and the SOM trained on ISOT

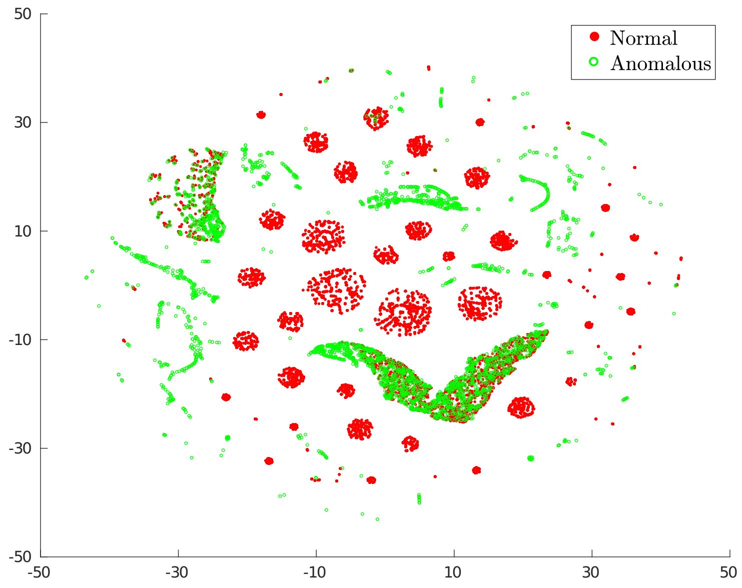


(a) tSNE mapping of CTU13a

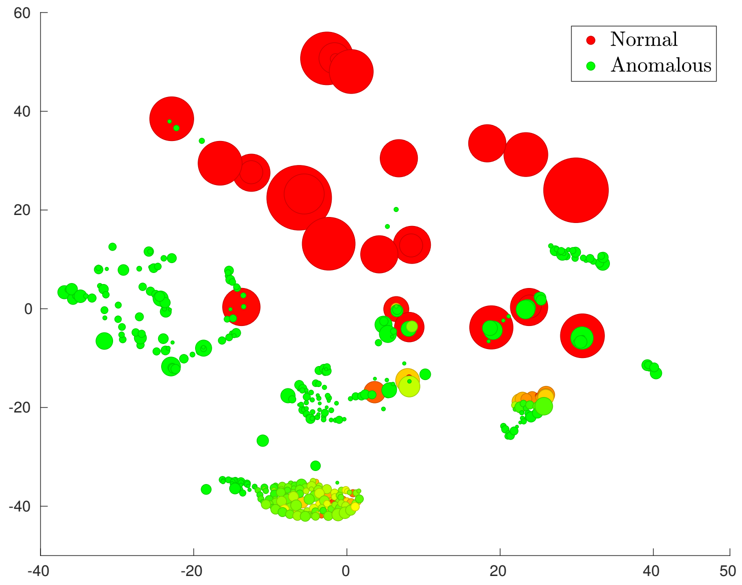


(b) tSNE mapping of the SOM trained on CTU13a. The size of each node denotes the number of data points that hit the node, while the mixed colour in a node denotes that the node is the BMU for data from more than one class.

Figure 4.2: tSNE mappings of CTU13a data set and the SOM trained on CTU13a



(a) tSNE mapping of HTTP-CSIC



(b) tSNE mapping of the SOM trained on HTTP-CSIC. The size of each node denotes the number of data points that hit the node, while the mixed colour in a node denotes that the node is the BMU for data from more than one class.

Figure 4.3: tSNE mappings of HTTP-CSIC data set and the SOM trained on HTTP-CSIC

between the mappings clearly demonstrate the ability of the SOM in summarizing and generalizing the input data. On visually comparing subplot (a) to the corresponding subplot (b) it is apparent that the SOM codebooks preserve the general structure of the data, while also reducing the noise and combining similar data regions into SOM nodes. Specifically, the distribution with which each class is represented is retained, as is the degree of mixing/purity with which different classes are expressed.

4.2 Experimental Settings

4.2.1 Data sets

From the original network traffic captures and the web access (request) logs, the data sets are exported as numerical vectors representing network flows or HTTP requests (see Section 3.1). Tables 4.2 and 4.3 present the flow and the request distributions in CTU13, ISOT, and HTTP-CSIC data sets, respectively. The botnet C&C and botnet attack traffic in CTU13 sets are distinguished as C&C and Botnet in the table. Due to the highly skewed distributions, the C&C flows and Botnet flows in CTU13 c-g and j-l data sets are considered one class (Botnet and C&C) in the experiments.

Given that the SOM is built based on the distances between data vectors and nodes, the features of the training data are normalized with zero means and unit variance before they are used for training.

Table 4.2: Data distribution in CTU13 sets

Scen.	Bot	# Packets	# Flows	Normal Flows	C&C Flows	Botnet Flows	Background Flows
a	Neris	71,971,482	2,824,637	30,387 (1.08%)	341 (0.01%)	40,620 (1.44%)	2,753,288 (97.47%)
b	Neris	71,851,300	1,808,123	9,120 (0.50%)	673 (0.04%)	20,268 (1.12%)	1,778,061 (98.34%)
c	Rbot	167,730,395	4,710,639	116,887 (2.48%)	63 (0.00%)	26,759 (0.57%)	4,566,929 (96.95%)
d	Rbot	62,089,135	1,121,077	25,268 (2.25%)	49 (0.00%)	1,719 (0.15%)	1,093,228 (97.52%)
e	Virut	4,481,167	129,833	4,679 (3.60%)	206 (0.16%)	695 (0.54%)	124,252 (95.70%)
f	Menti	38,764,357	558,920	7,494 (1.34%)	199 (0.04%)	4,431 (0.79%)	546,795 (97.83%)
g	Sogou	7,467,139	114,078	1,677 (1.47%)	26 (0.02%)	37 (0.03%)	112,337 (98.47%)
h	Murlo	155,207,799	2,954,231	72,822 (2.47%)	1,074 (0.04%)	5,053 (0.17%)	2,875,281 (97.33%)
i	Neris	115,415,321	2,087,509	29,967 (1.44%)	2,973 (0.14%)	182,014 (8.72%)	1,872,554 (89.70%)
j	Rbot	90,389,782	1,309,792	15,847 (1.21%)	37 (0.00%)	106,315 (8.12%)	1,187,592 (90.67%)
k	Rbot	6,337,202	107,252	2,718 (2.53%)	3 (0.00%)	8,161 (7.61%)	96,369 (89.85%)
l	NSIS.ay	13,212,268	325,472	7,628 (2.34%)	25 (0.01%)	2,143 (0.66%)	315,675 (96.99%)
m	Virut	50,888,256	1,925,150	31,939 (1.66%)	536 (0.03%)	39,467 (2.05%)	1,853,207 (96.26%)

Table 4.3: Data distribution in ISOT and HTTP-CSIC data sets

ISOT	**# Packets**	**# Flows**	**# Normal flows**	**# Storm flows**	**# Waledac flows**
	25,284,617	264,882	212,203 (80.11%)	18,721 (7.07%)	33,958 (12.82%)
HTTP-CSIC	**# Requests**		**# Normal req.**	**# Malicious req.**	
	71,485		56000 (78.34%)	15,485 (21.66%)	

4.2.2 Performance measurements

Except where noted, all results are analyzed with respect to 20 independent trials in *three fold cross validation* in order to ensure statistical significance. The training and testing partiontion sizes for each data set are presented in Appendix B.1.

For the classification tasks, the results are measured in terms of: Accuracy, Class-wise Detection Rate (CDR), Class Detection Rates (DRs), and Precision of malicious classes. While Accuracy and Detection rates show the system's capability in correctly classifying test instances into classes, the Precision denotes the percentage of raised alerts that is accurate. In security applications, high precision classification systems will reduce the amount of false alerts that require attention from the network administrators. The measurements and the data distributions will be presented in % in the following sections.

$$Accuracy = \frac{\text{Number of correctly classified test instances}}{\text{Total number of test instances}}. \tag{4.1}$$

$$DR_i = \frac{\text{Number of correctly classified class } i \text{ test instances}}{\text{Total number of class } i \text{ test instances}}. \tag{4.2}$$

$$CDR = \frac{1}{N_c}\sum_i DR_i, \text{ where } N_c \text{ is the number of classes.} \tag{4.3}$$

$$Precision = \frac{\text{Number of correctly classified malicious test instances}}{\text{Total number of test instances classified as malicious}}. \tag{4.4}$$

4.3 SOM Classification Results

4.3.1 Training scheme (i)

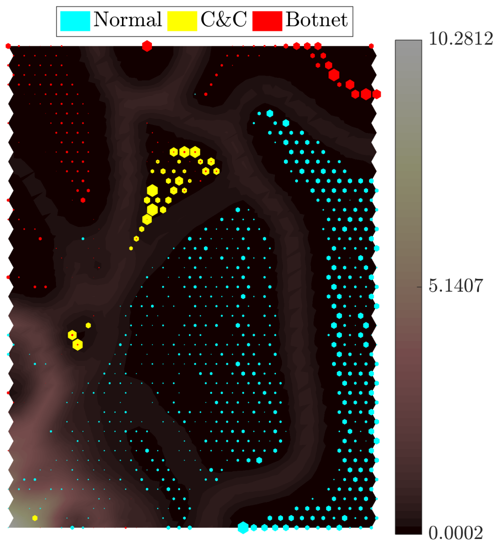
Table 4.4: Classification performance of SOM training scheme (i)

Data set	**Accuracy**	**CDR**	**Detection rate**			**Precision**
CTU13			**Normal**	**C&C**	**Botnet**	
a	98.15	97.51	99.25	95.92	97.35	99.45
b	96.57	95.67	99.36	92.18	95.46	99.72
c	99.66	99.77	99.59	99.95		98.25
d	99.54	99.66	99.51	99.82		95.38
e	98.86	99.25	98.68	99.82		93.58
f	99.22	99.35	98.79	99.92		98.08
g	97.20	97.78	97.16	98.41		56.75
h	99.55	99.46	99.56	99.34	99.49	95.43
i	96.43	97.32	99.15	96.84	95.96	99.87
j	99.81	99.46	98.98	99.94		99.85
k	99.73	99.56	99.23	99.90		99.74
l	99.19	99.38	98.90	99.85		97.49
m	96.73	93.30	99.40	85.72	94.79	99.52
ISOT			**Normal**	**Storm**	**Waledac**	
	95.31	93.69	95.91	91.38	93.77	85.71
HTTP-CSIC			**Normal**	**Malicious**		
	92.81	93.67	92.42	94.91		69.77

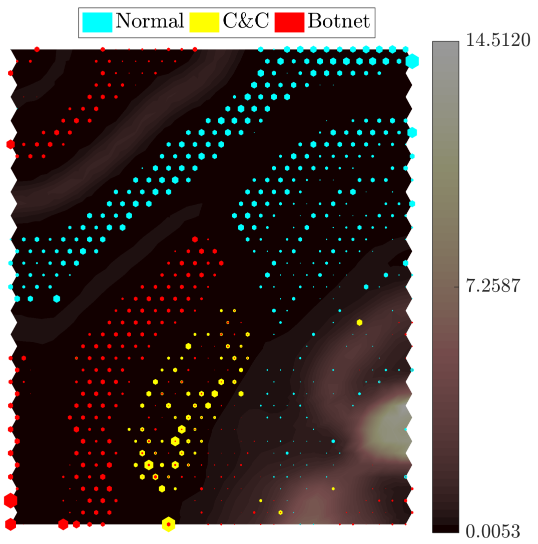
Table 4.4 presents in detail the classification performance of the SOMs trained using scheme (i), using both normal and malicious data for training the SOMs. The results are obtained on the unseen test partitions of the data sets. As it shows in the table and Figures 4.4 and 4.5, this training scheme achieves high performance with a clear separation between non-overlapping groups of BMUs of traffic classes on the hit maps. For example, in Figures 4.4, which visualize data distribution of different

Table 4.5: Classification performance of SOM training scheme (i) when only 2 classes are taken into account

Data set	**Accuracy**	**CDR**	**DR Normal**	**DR Botnet**	**Precision**
CTU13a	99.64	99.59	99.25	99.93	99.45
CTU13b	99.66	99.58	99.36	99.80	99.72
CTU13h	99.59	99.74	99.56	99.92	95.43
CTU13i	99.74	99.49	99.15	99.83	99.87
CTU13m	99.69	99.66	99.40	99.92	99.52
ISOT	96.48	97.36	95.91	98.80	85.71

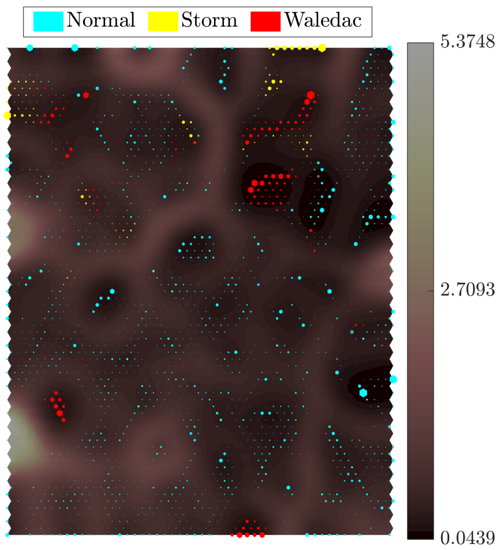


(a) CTU13a

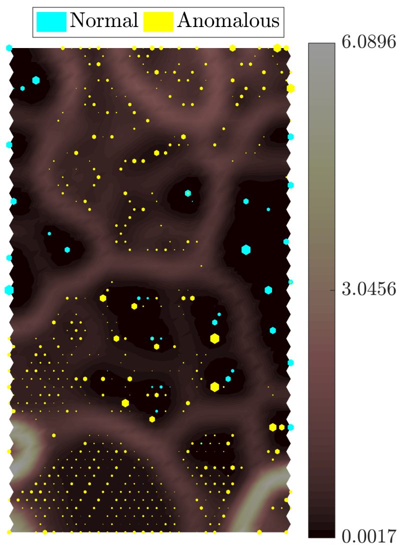


(b) CTU13m

Figure 4.4: Hit maps of two CTU13 data sets on the SOMs trained using scheme (i)



(a) ISOT



(b) HTTP-CSIC

Figure 4.5: Hit maps of ISOT and HTTP-CSIC data sets on the SOMs trained using scheme (i)

traffic classes in CTU13a and CTU13m on the SOMs, it is clear that the different classes are either separated by a lighter area in the SOM Umatrix, which indicates large inter-node distances, or empty nodes.

Figures 4.5(a) and (b) shows hit maps of the SOMs trained on ISOT and HTTP-CSIC data sets, respectively. Although the separation between legitimate and malicious classes are not as obvious as that of the CTU13 sets, one can still easily distinguish different traffic classes by looking at the hit maps. This supports the hypothesis for the ability of the SOM to model network behaviours, and separate malicious traffic from normal traffic.

Moreover, in the CTU13 sets with separate classes for C&C and Botnet attack traffic, most of the incorrectly classified Botnet (C&C) flows are still classified as C&C (Botnet). The same observation can be made between Waledac and Storm botnet traffic in ISOT data set as well. Table 4.5 presents the classification performances of the trained SOMs using scheme (i) when only two classes (Normal and Malicious) are taken into account. This also shows the flexibility of the SOM in particular and unsupervised learning in general in interpreting the learned model on the data. By not using the ground-truth in the learning process, the system instead learns to generalize the input data, and provides the ability to obtain results based on specific interests.

Finally, it should be noted that the SOMs trained using both normal and malicious data achieved comparable results to that of supervised learning systems in the papers where the ISOT and HTTP-CSIC data sets are introduced. On ISOT data set, the detection rates of REPTree classifier with reduced subset in [59] were 97.9% and 98.1% for Normal and Malicious flows, respectively. On the HTTP-CSIC data set, the average accuracy and Normal DR obtained using decision tree learning algorithms were 93.65% and 93.1%, respectively [39].

4.3.2 Training scheme (ii) Normal only and (iii) Malicious only

Table 4.6: Classification performance of SOM training scheme (ii) Normal only

Data set	τ	**Accuracy**	**CDR**	**Normal DR**	**Malicious DR**	**Precision**
CTU13a	0.90	38.64	45.12	88.25	2.00	18.15
CTU13b	0.95	58.42	68.05	92.56	43.55	92.26
CTU13c	0.95	94.82	96.59	93.76	99.42	78.54
CTU13d	0.93	91.55	87.43	92.49	82.37	51.63
CTU13e	0.95	87.84	79.09	92.02	66.15	59.70
CTU13f	0.94	94.09	94.72	92.06	97.39	88.36
CTU13g	0.94	88.52	79.55	89.22	69.88	19.79
CTU13h	0.93	86.61	63.17	91.34	35.01	26.35
CTU13i	0.93	38.05	59.76	89.56	29.96	90.69
CTU13j	0.95	63.32	75.50	91.95	59.05	97.82
CTU13k	0.95	93.88	93.33	92.23	94.42	97.33
CTU13l	0.95	85.64	81.62	91.61	71.63	78.40
CTU13m	0.95	68.73	71.28	93.35	49.21	88.79
ISOT	0.93	75.76	52.96	90.82	15.10	29.04
HTTP-CSIC	0.94	90.01	81.21	93.99	68.42	68.02

The testing results of the SOMs trained using only normal data and only malicious data are presented in Tables 4.6 and 4.7, respectively. The ROCs of classification systems based on SOMs trained using the two schemes are shown in Figures 4.6 and 4.7. Figures showing hit maps of testing sets are presented in Appendix B.2.

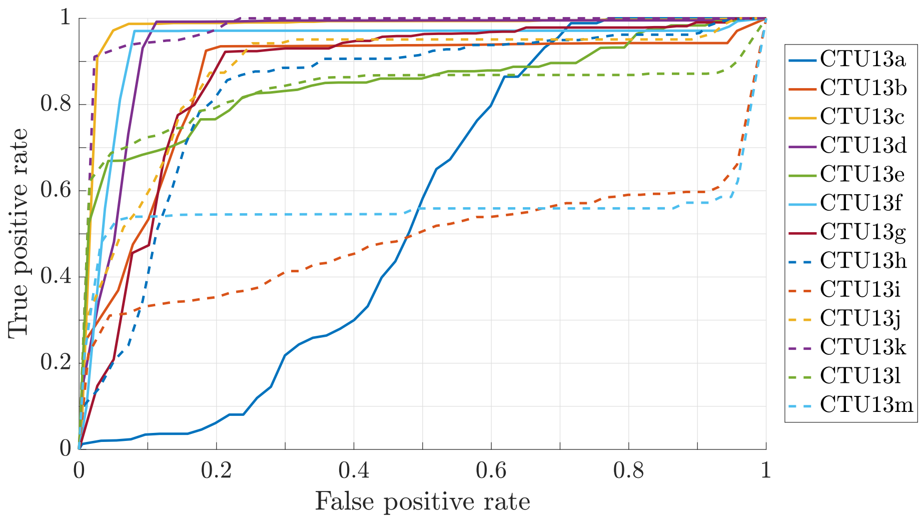
Table 4.7: Classification performance of SOM training scheme (iii) Malicious only

Data set	τ	**Accuracy**	**CDR**	**Normal DR**	**Malicious DR**	**Precision**
CTU13a	0.94	83.02	81.34	70.17	92.52	83.90
CTU13b	0.90	61.90	44.71	1.01	88.41	67.22
CTU13c	0.95	92.31	92.06	92.46	91.65	82.85
CTU13d	0.93	57.38	71.88	54.09	89.67	21.83
CTU13e	0.90	26.01	48.45	15.30	81.60	16.74
CTU13f	0.94	66.69	71.25	51.94	90.56	55.70
CTU13g	0.93	50.11	63.00	49.10	76.90	14.53
CTU13h	0.95	51.72	70.10	48.02	92.19	16.26
CTU13i	0.95	93.44	92.23	90.57	93.90	98.48
CTU13j	0.92	82.50	58.68	26.52	90.84	89.26
CTU13k	0.94	86.49	81.72	72.18	91.25	91.20
CTU13l	0.95	56.70	66.62	41.96	91.27	40.56
CTU13m	0.90	50.97	46.04	3.30	88.78	53.65
ISOT	0.95	65.39	75.16	58.93	91.39	39.54
HTTP-CSIC	0.94	45.49	64.71	36.80	92.62	21.62

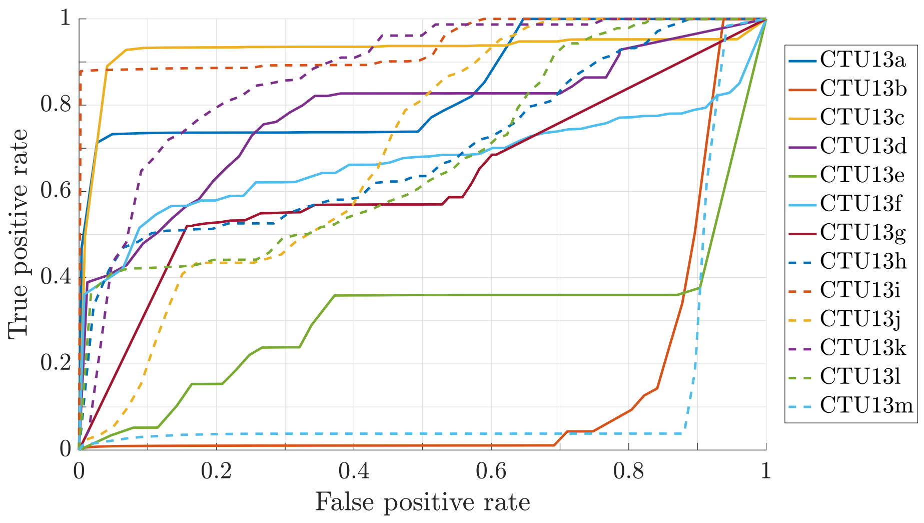
Among the two training schemes, the results are generally better with the scheme using Normal data only. Ensuring the False positive rates no greater than 8%, the scheme (ii) is typically able to detect more than 60% of malicious data vectors. On the other hand, the training scheme using only Botnet traffic observes poor performance on most of CTU13 data sets and HTTP-CSIC.

On ISOT data set, the trend is reversed, where SOM training by only botnet flows gives far better results than the SOM trained by normal data only. However, considering that ISOT data set is a combination of normal and malicious data provided by different organizations, this might be the reason why the trend is reversed. The normal traffic in ISOT data set was provided by LBNL and the malcious traffic was captured using Honeypots [59]. So, the results are based on data captured at different organizations (networks) under (potentially) different topologies and conditions.

Another observation that can be made based the scheme (ii) results on CTU13 sets is that the performance suffers when the amount of training data is not sufficient, e.g. on CTU13 a,b,m. This suggests that expanding the normal training set to cover more normal behaviours and protocols can improve the results obtained using scheme (ii).

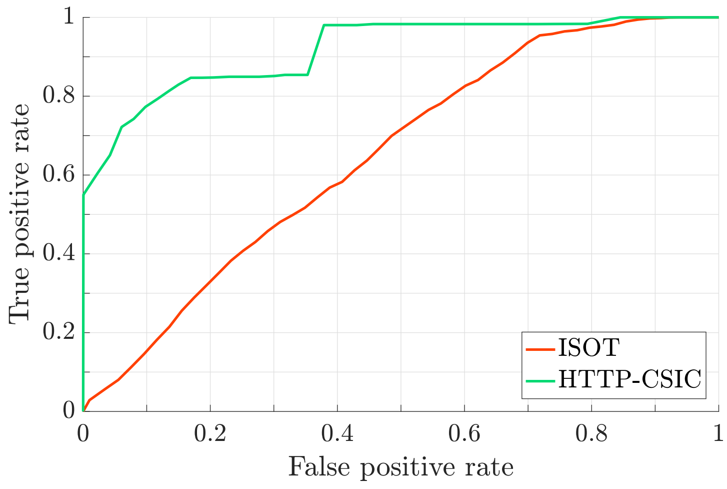


(a) CTU13 sets, training scheme (ii) Normal only

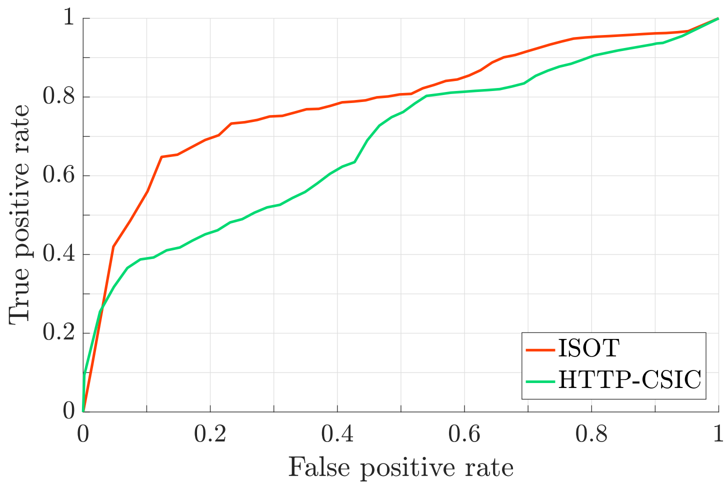


(b) CTU13 sets, training scheme (iii) Malicious only

Figure 4.6: ROCs of classification systems based on SOMs trained using schemes (ii) and (iii) on CTU13 data sets

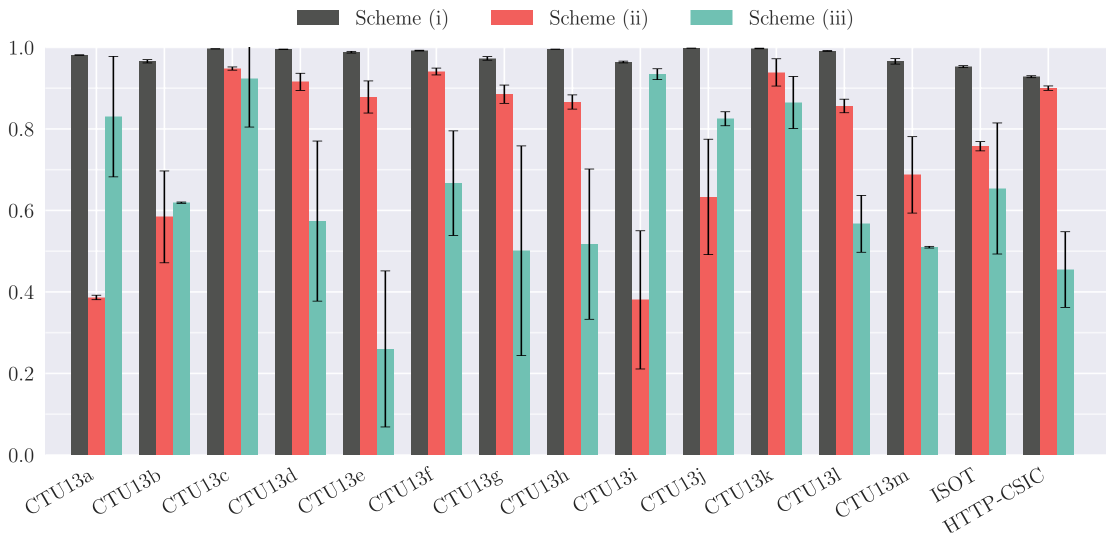


(a) ISOT and HTTP-CSIC, training scheme (ii) Normal only

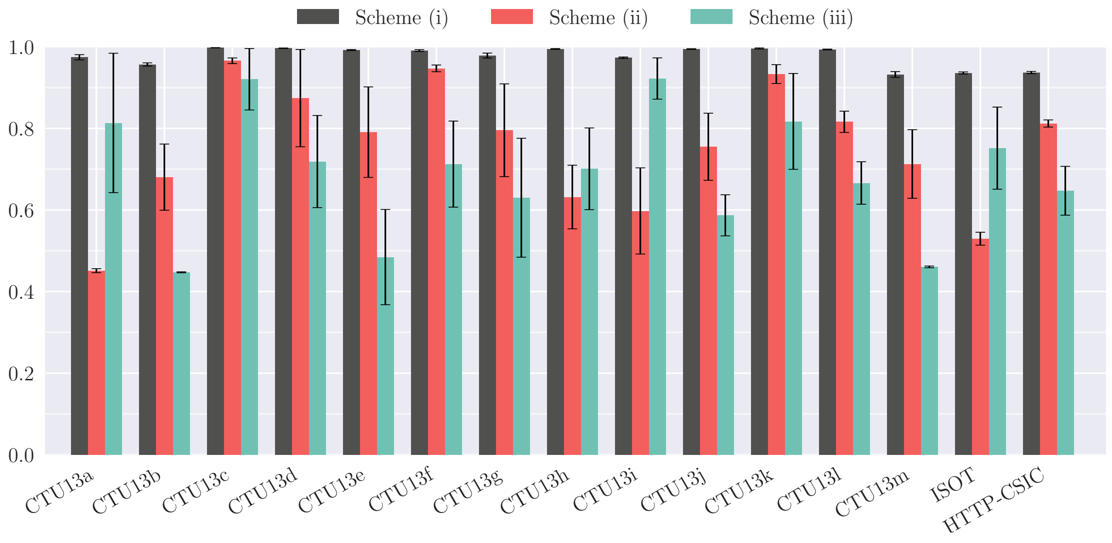


(b) ISOT and HTTP-CSIC, training scheme (iii) Malicious only

Figure 4.7: ROCs of classification systems based on SOMs trained using schemes (ii) and (iii) on ISOT and HTTP-CSIC data sets



(a) Accuracy



(b) Class-wise Detection Rate

Figure 4.8: Summarization of classification performance of the three SOM training schemes. The columns represent average values, while the errorbars correspond to one standard deviation.

4.3.3 Discussion

The summary of the performances of the three SOM training schemes are presented in Figure 4.8. As expected, SOM training scheme using both normal and botnet traffic / malicious requests achieves the highest performance. Hence, for higher accuracies, data analytics systems trained on both malicious and normal behaviours should be preferred. The results also suggest that when a complete set of training data is not available, SOMs can be trained on well-identified normal data only and still achieve a reasonable performance, given that the data is diverse enough to cover most of the legitimate traffic.

Another observation from the experiments is that C&C and botnet attack traffic (in the CTU13 data sets) are relatively different. The examples of this can be seen in Figures 4.4(a) and (b), where the two malicious classes distributed over separated regions in the hit maps. This might be based on the essence of these two traffic types. While flows labelled Botnet represent attacks and malicious activities, C&C flows are for maintaining the botnet and issuing attack orders. Thus, naturally the hackers would want to conceal the C&C traffic to make it as similar to the normal traffic as possible. The observation suggests that independent investigation strategies for Botnet and C&C traffic may improve detection systems' performance.

4.4 SOM vs Other Learning Algorithms

In this section, the performance of the model proposed in this thesis is benchmarked against other learning techniques, both unsupervised and supervised, from the literature. Four selected algorithms are: C4.5, Naive Bayes, X-means clustering, and EM, in which the former two are supervised, while the latter two are unsupervised. The chosen algorithms are amongst the most popular ones in data mining, as stated in [56]. To comply to the training data requirements of these algorithms, only training sets consisting of both normal and malicious data (as in scheme (i)) are employed. The implementations of the benchmarking algorithms are based on Weka [23].

4.4.1 Learning algorithms for comparison

C4.5

C4.5 is an algorithm to generate decision tree for classification, which is extended from the earlier ID3 algorithm developed by Ross Quinlan [44]. C4.5 uses the concept of *information entropy* for creating if-then rule set at each tree node in order to build the tree. The training data is required to be completely labelled and contain at least two classes. At each node of the tree, the data is split into subsets which contains only one or a few classes as majority. The criteria is satisfied most effectively by choosing the attribute and the split point that gives the highest normalized information gain. The C4.5 algorithm then recurs on the subtree. More detailed information on the C4.5 learning algorithm can be found in [44]. For the parameters, Weka's default C4.5 parameters are applied. Only the minimum number of instances per leaf are adjusted according to the training data size.

Naive Bayes

Naive Bayes classifier is a simple probabilistic classifier based on Bayes' theorem, in which strong independence is assumed between the data attributes. From the training data, two probabilities can be calculated using Bayes theorem: the probability of each class, and the conditional probability for each given training sample. Then, for each input (testing) example, the posterior probability for each class are calculated using Naive Bayesian equation to act as a vote toward the class. More details on the algorithm can be found in [3].

X-means

X-means clustering, an extended K-Means algorithm [35] which simplifies the deployment of the latter by automatically determining the number of clusters [41], is also employed for benchmarking in this thesis. Essentially K-means is a method of vector quantization, which is in the same group as the SOM algorithm, that aims to partition input observations in k clusters by the distances to the closest clusters' means. Starting from K-means with a small number of clusters, X-means employs the Bayesian Information Criterion concept to make local decisions about which subset of current centroids should be split. A more detailed description of X-means learning algorithm can be found in [41].

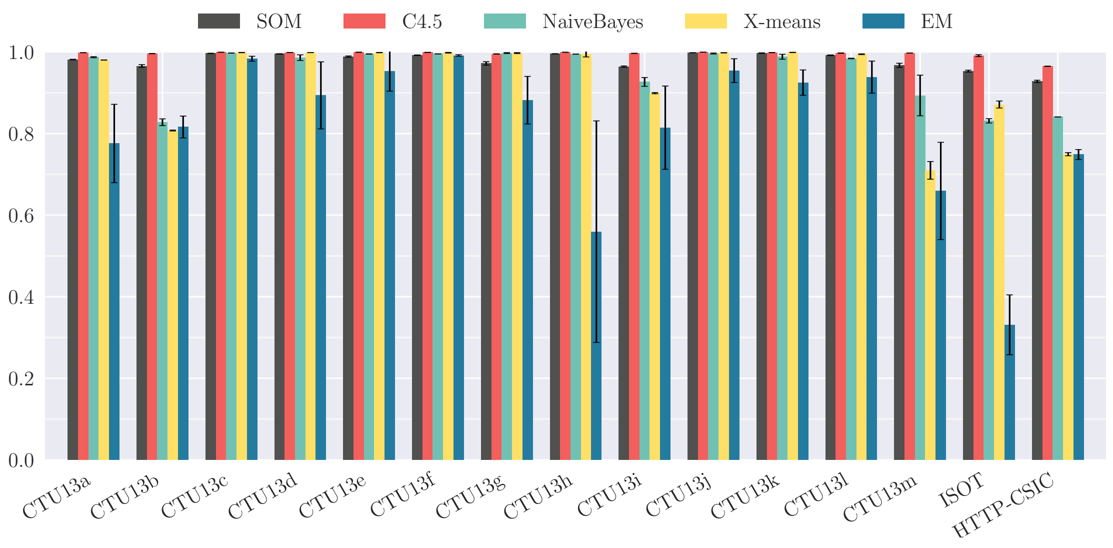
EM

The Expectation Maximization (EM) algorithm is an iterative method for fitting normal mixture models by maximum likelihood [11]. The algorithm estimates the model parameters by alternating between two steps: expectation (E), and maximization (M). In the E step, a function is created for the expectation of the log-likelihood obtained using the current estimate for the parameters. Then in the M step, the expectation function found in the E step is maximized in order to compute the parameters. Iteratively, these estimated parameters are then used as the input for the next E step. More details on the algorithm can be found in [3]. In the evaluations done in this thesis, the number of clusters for EM algorithm is determined using the cross validation option in Weka [23].

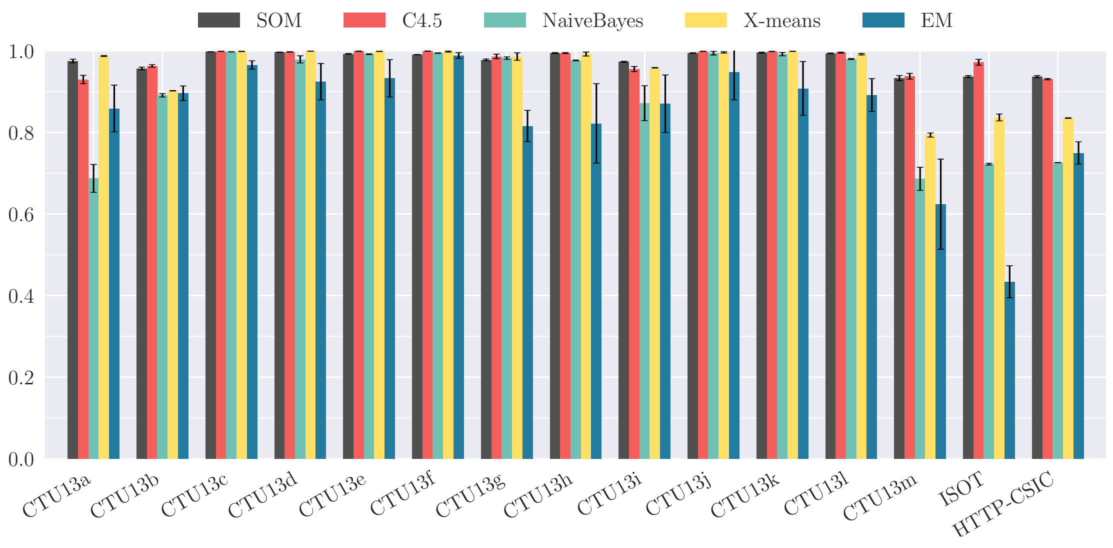
4.4.2 Results

The classification performances of the five algorithms on CTU13, ISOT, and HTTP-CSIC data sets are summarized in Table B.2 and Figure 4.9. It is clear that the SOM outperformed EM and even Naive Bayes, which is a supervised learning algorithm. In comparison with X-means, the SOM is comparable on most of CTU13 data sets, and outperformed on CTU13b,i,m, ISOT, and HTTP-CSIC. The results of SOMs are also comparable to C4.5, and even better in CDR for CTU13a, CTU13i and HTTP-CSIC. Statistical support for the analysis of the results is obtained by the hypothesis testing techniques, as presented in Appendix B.3.1.

The advantages of the proposed system are not only in the classification performance, but also in the ability to visualize the data. For example, when ground-truth is available, one can analyze SOM hit maps (Figure 4.4) to identify the regions, and hence the portion of the data, that may require further analysis. On the other hand, when the ground-truth is not available, proposed approach is still able to provide the expression on the structure of input data. Finally, in the case where training data is incomplete, it is possible to train the SOM in one class fashion to work as an outlier detector as in 4.3.2, while it is more challenging to do the same with the



(a) Accuracy



(b) Class-wise Detection Rate

Figure 4.9: Summarization of classification performance of five algorithms. The columns represent average values, while the errorbars correspond to one standard deviation.

aforementioned algorithms.

4.4.3 Adaptability of algorithms on novel malicious behaviours

To examine the ability of the algorithms on detecting novel (new / unseen during training) malicious behaviours, one experiment with special training and testing sets is carried out. The training / testing split is done in the same manner as in [15], where 5 sets (a,b,f,h,i) in CTU13 are used for training, while the rest (CTU13c-e, g, j-m) are used for testing. The split ensures that the testing set contains only novel botnets (Rbot, Virut, Sogou, NSIS.ay) that are not included in the training set (Neris, Menti, Murlo). Furthermore, while the botnets in the training set are only IRC botnets, or use a proprietary protocol for their C&C communication, the botnets in testing set establish their connections on not only IRC but also HTTP/HTTPS and P2P protocols (Table 3.1). The SOM and other algorithms are trained using both normal and botnet traffic flows in the training set (as in scheme(i)). Due to the training data size, only 25% or the original training data is randomly sampled to train the algorithms. The trial is repeated 20 times for each algorithm.

Table 4.8: Classification performances of five algorithms on CTU13 test set containing unseen botnet

Algorithm	Accuracy	CDR	Normal DR	Botnet DR	Precision
SOM	**98.17**	**98.11**	99.26	**96.96**	99.16
C4.5	83.54	82.68	**99.89**	65.48	**99.82**
Naive Bayes	72.05	70.70	97.82	43.59	94.75
X-means	81.88	81.01	98.59	63.42	97.61
EM	74.54	73.44	95.58	51.29	92.33

The results are presented in Figure 4.10 and Table 4.8. It is apparent that the SOM was able to retain its good performance on the novel botnets, while the algorithms for comparison failed. This demonstrates the potential of the proposed approach in dealing with unseen behaviours, even when new communication protocols are employed for botnet C&C, by reducing the dependence on *a priori* ground-truth in the learning process.

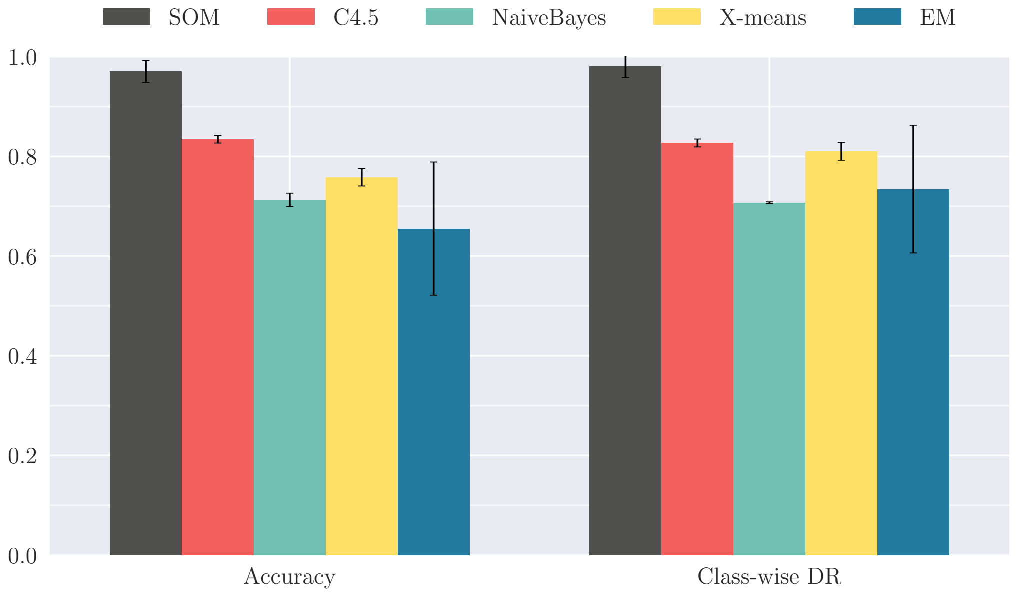
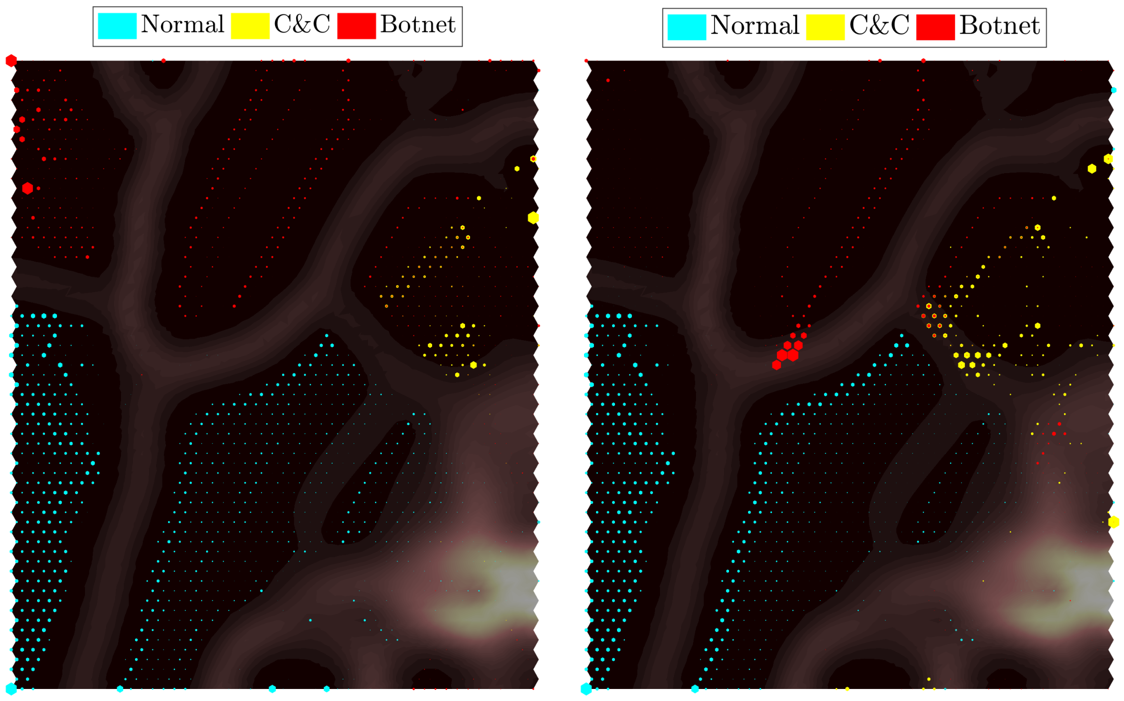


Figure 4.10: Accuracy of 5 algorithms on CTU13 novel botnets testing set



(a) Hit map of training data

(b) Hit map of testing data

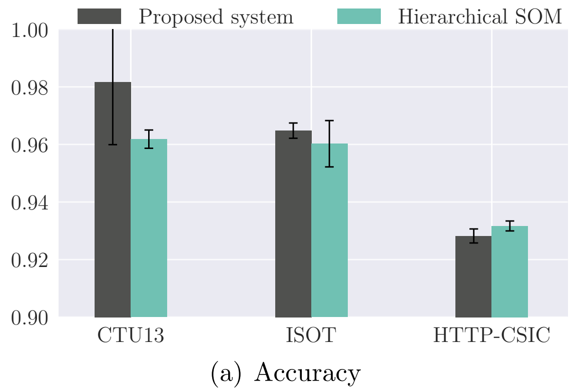
Figure 4.11: Hit maps of training and testing data, novel botnet detection experiment

To further analyze the SOMs in this experiment, the hit maps for training and testing data are plotted in Figure 4.11. As it is shown in the figure, although the C&C and botnet attack behaviours in the testing set are somewhat different from the botnets in the training set, and appears to be more similar (closer in the hit maps) to the normal traffic, the SOM was still able to successfully identify them. The figures also suggest that the shifting in network behaviours should be gradually incorporated (by re-training) into the system to maintain a good detection capability. This can be done by periodically training the SOM at regular intervals, or re-training when new network behaviours are noticed.

4.5 Comparison with Hierarchical SOM

In this section, the performance of the proposed system based on one layer of SOM is compared against methods that employ multiple layers of SOM. The hierarchical SOMs employed for comparison is similar to the system proposed in [30]. In the system, the first layer consists of a 6 × 6 SOM. Post training, the SOM is analyzed to identify the nodes with mixed hits from multiple classes. The best matching data points for each such node are then used to build (train) a second layer of SOM. The system can recur until the third layer of SOMs are built. Inherently, this training process make the hierarchical SOM more dependent on the data labels. This type of system can be termed semi-supervised.

As it is shown in Figure 4.12, the proposed system in this thesis achieves slightly better performances on CTU13 (as in 4.4.3) and ISOT, and comparable results on HTTP-CSIC. The performances of the two approaches on the individual CTU13 data sets are not shown due to exceeding similarities. Wilcoxon signed-ranks tests are performed on Accuracy and CDR results of the two approaches on the data sets. The obtained p-values for Accuracy and CDR are 0.50 and 0.47, respectively. Thus one can not reject the null-hypotheses that the two approaches achieve the same performances. This confirms that SOMs with sufficient resolution could accurately model the input data for attack behaviours discovery, and also eliminates the need of using data labels, which may be not available, during the training phase.



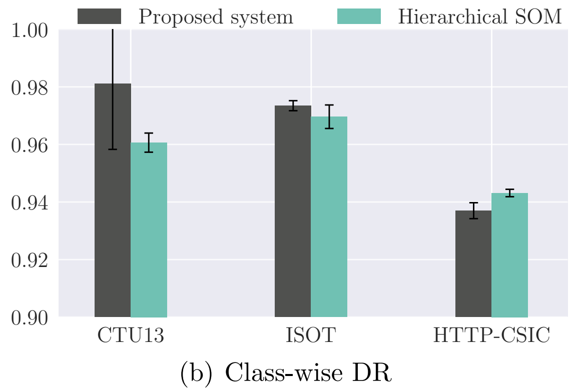
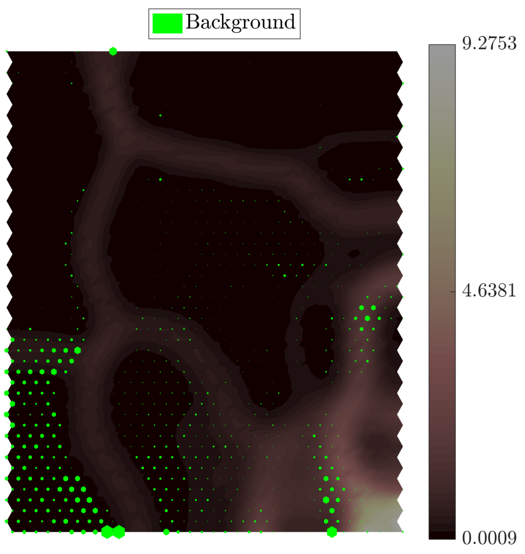


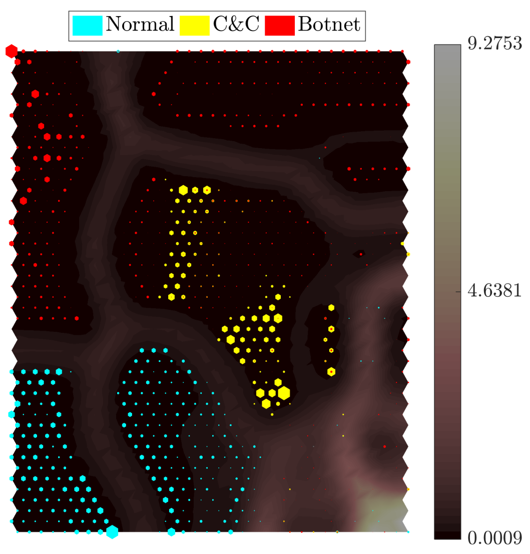
Figure 4.12: Performances of proposed system and Hierarchical SOM

4.6 SOM for Analyzing Unknown Data

In the this section, the trained SOMs presented in Section 4.3 are used to analyze the distribution of Background (unlabelled / unknown) data in the CTU13 sets. Appendix B.4 presents in detail the distribution of Background traffic flows on the trained SOMs.



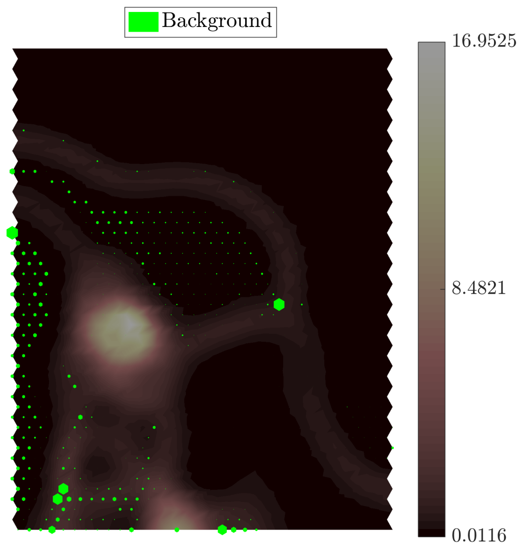
(a) Background flows, CTU13i



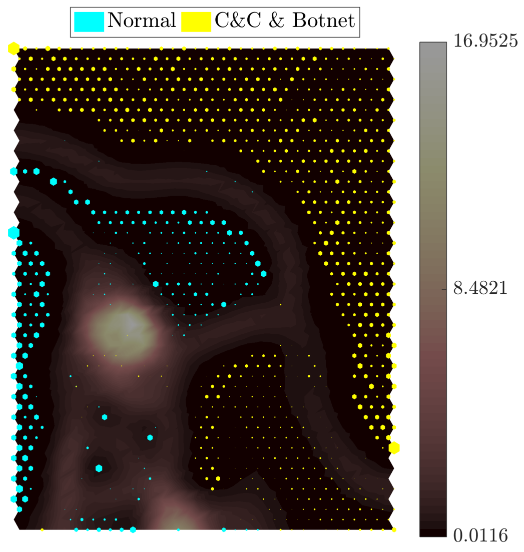
(b) Normal, C&C and Botnet flows, CTU13i

Figure 4.13: Hit maps of CTU13i Background (left) and training data (right) on the SOM trained using scheme (i)

Since there is neither ground-truth nor packet capture payload provided by the



(a) Background flows, CTU13j



(b) Normal, C&C and Botnet flows, CTU13j

Figure 4.14: Hit maps of CTU13j Background (left) and training data (right) on the SOM trained using scheme (i)

CTU for this portion of the data sets, the SOMs trained using scheme (i) are employed as the baseline to analyze the unknown data portion of the CTU13 data set, backed by the promising results obtained in the previous sections. As shown in Figures 4.13, 4.14, and Table B.4, the proposed system identifies most of the Background flows as Normal / Legitimate (53%-67%, depending on the CTU13 data set analyzed), and only a small portion (0 - 6%) as possibly Botnet traffic. On the other hand, the rest of the Background traffic flows appear to be very different from both Legitimate and Botnet/C&C. The proposed system suggests that these flows are labelled as anomalies for further investigation. Manually inspecting the background flows labelled as Anomaly, it is found that many of them have unfamiliar protocols that were not seen in the training data, for example Address Resolution Protocol, Real-time Transport Protocol (RTP), RTP Control Protocol, and Internet Group Management Protocol. This intuition suggests that the training sets need to be expanded to cover more behaviours and protocols.

Figures 4.15 and B.4 show the statistics in Table B.4 for comparison. SOMs

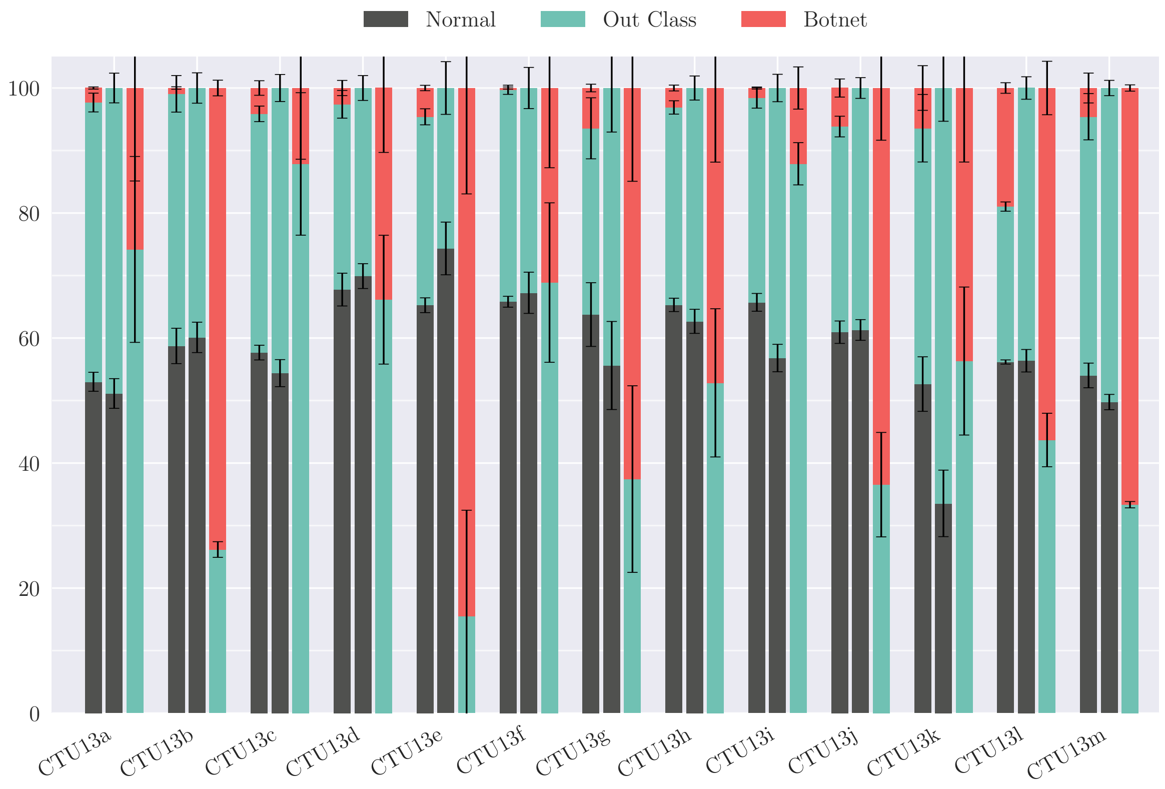


Figure 4.15: Stacked bar chart showing distribution of CTU13 Background traffic flows. For each data set, the three columns show distribution of Background flows on SOMs trained using scheme (i), (ii), and (iii) respectively. Each component in a column represent average values, while the errorbars correspond to one standard deviation of that component. Out Class represents the partition of Background data that is different from the class(es) in training data.

trained by only Normal data (scheme (ii)) show very similar Background data distributions to scheme (i). The two schemes agree not only in the Normal flow distributions but also in which flows are labelled as Normal. On average, 84.8% (sd 4.14) of the Background flows labelled as Normal by the SOMs trained using scheme (ii) are also labelled as Normal by the SOMs trained using scheme (i), while only 1.92% (sd 1.81) is identified as Botnet by scheme (i) SOMs. On the other hand, SOMs trained by only Botnet/C&C data (scheme (iii)) label most of the Background flows as Botnet, with completely different distributions from what obtained from scheme (i). Specifically, only 7.06% (sd 6.17) of Background flows labelled as Botnet by scheme (iii) SOMs is confirmed by scheme (i) SOMs, while 59.07% (sd 18.08) of those flows is identified as Normal by scheme (i) SOMs.

To further investigate the Background traffic, we calculate the average quantization error (see 3.2.2) for each identified class (Normal, Botnet, Anomaly) of the Background traffic. For the sake of simplicity, this calculation is done on the training, testing data and the trained SOMs from 4.4.3. The quantization error ranges for labelled Background flows by the three training schemes are shown in Figure 4.16. Using scheme (i) trained SOMs, the average quantization errors of flows labelled as Normal is 1.06, while it is 0.42 for Botnet flows. These low quantization errors demonstrate that SOMs trained using scheme (i) label the Background traffic as Normal and Botnet with high confidence, considering that the overall average quantization error is 4.69. On the other hand, for the flows labelled as Anomaly, the average quantization error is 10. This higher value confirms the observation that anomaly traffic contains very different behaviours / patterns that were not present in the training data. This is further confirmed by the manual analysis of these flows and the different protocols identified as a result of this analysis. Similarly, SOMs trained using scheme (ii) give average quantization errors of 2.26 and 5.83 for flows classified as Normal and Anomaly. On the other hand, training scheme (iii) produces SOMs with much higher quantization errors when applied on the Background traffic. On average, the Background flows are classified as Botnet and Not Botnet with quantization error values of 23.74 and 7.63, respectively. These very high error values indicate that SOMs trained using scheme (iii) are not suitable for Background / unknown data analysis.

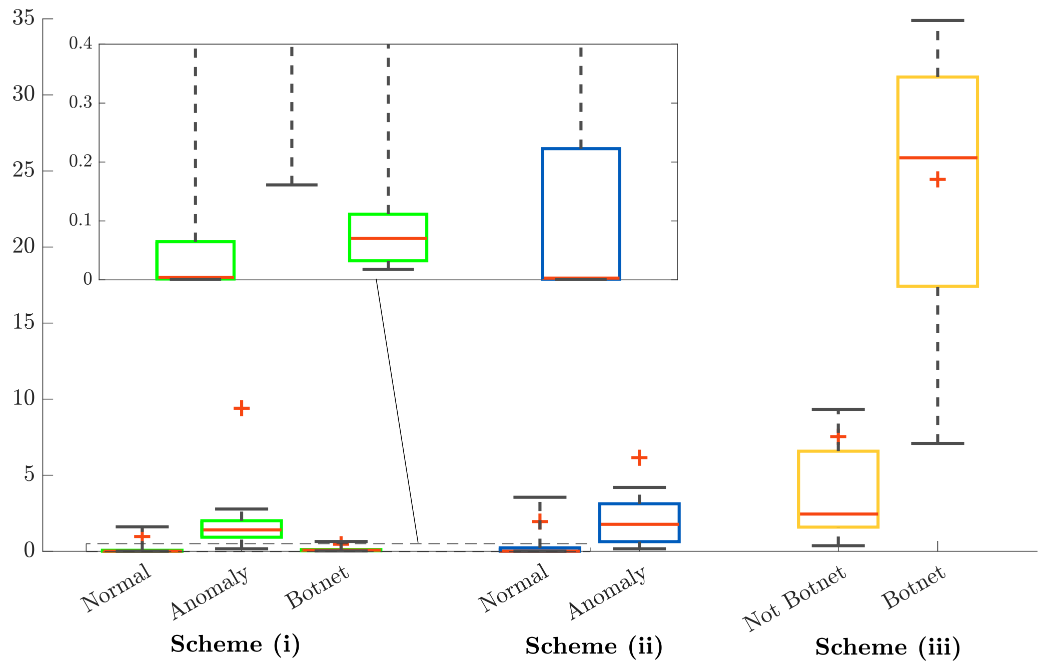


Figure 4.16: Quantization error ranges of Background flows by labels assigned using SOMs from three training schemes. The box represents the interquartile range, while the whiskers extend to the 9th percentile and the 91st percentile. Red line and plus sign shows the median and mean, respectively.

Chapter 5

Conclusion and Future Work

The main objectives in this thesis are: (i) investigating the capability of SOMs as an unsupervised data analytics system for modelling and classification of network behaviours, and (ii) using this capability for analyzing unknown/unlabelled traffic. Specifically, the thesis shows the capability of such an approach without scrutinizing network packet content. Thus, this enables a simple solution that is more flexible and adaptable to different deployment conditions and environments.

Using three different SOM training schemes, the capabilities of this SOM based approach are analyzed and evaluated on publicly available data sets of modern botnets and web attacks .The obtained results are comparable to that of previous supervised machine learning-based approaches, even though the proposed approach is based on the unsupervised learning paradigm. Detection rates of Botnet and Normal classes are up to 99.95% and 99.59% with the training scheme using both classes. Moreover the experiments on unseen botnets and unknown traffic portions show the potential of the approach for building a strong data analytics system for unknown traffic analysis. Our data analytics results on unknown traffic also suggest that when a complete set of training data is not available, SOMs can be trained on normal data only and still achieve a competitive level of performance, given that the data is diverse enough to cover most part of the legitimate traffic.

Multiple directions are promising for extending the thesis research. First, the process of determining the SOM size can be done based on the characteristics of input data instead of current heuristic method. One example of this approach is using Gap Statistics for determining the number of internal data clusters, and use that to determine the number of SOM units needed.

Feature learning plays an important role in machine learning solutions. Given that the SOM building process is based on distance measures, feature extraction and selection methods can be applied to reduce the unnecessary linear dependency between the input features, and highlight the important attributes for discriminating the malicious behaviours. Moreover, this may ease the curse of dimensionality, and hence introduce the approach to more potential applications. However, preliminary experiments using PCA for the task do not show any sign of improving performances or significantly reducing training time. Hence, one may attempt to adapt nonlinear dimensionality reduction methods for the preprocessing task, e.g. kernel PCA, or bottleneck neural network. On the other hand, although the approach performed well on the CTU13 data set with only basic flow attributes, detailed feature analysis can be conducted to reveal the effect of such limited feature set on the performance and the generalization ability of the approach, as well as to investigate how informative each flow feature is in terms of traffic classification.

More analysis in the combined use of SOM hit counts and quantization errors as a filter for unseen data can be carried out. This would test the ability of such an approach in reducing the noise in data and increasing the accuracy. Furthermore, probability models, such as the model in [2], can be applied for effectively confirming if a new data sample belongs to the SOM's trained data distribution or not.

Finally, the performance of the SOM-based data analytics system can be studied against other data sets, to examine its potential of detecting other types of network attacks and malicious activities. One solution is to generate a more realistic data set that captures real-time traffic and real-world security situations. From that, the method can also be improved to deal with live training data and live detection scenarios. The generalization ability of the approach can also be analyzed in more extensive studies, similar to Section 4.4.3, where the model have to deal with detection of unseen and unmodelled threats appearing in the traffic post training.

Bibliography

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, pages 487–499, San Francisco, CA, USA, 1994. Morgan Kaufmann Publishers Inc.

[2] E. Alhoniemi, J. Himberg, and J. Vesanto. Probabilistic measures for responses of Self-Organizing Map units. In *Proceedings of the International ICSC Congress on Computational Intelligence Methods and Applications (CIMA'99)*, pages 286–290. ICSC Academic Press, 1999.

[3] E. Alpaydin. *Introduction to machine learning*. MIT Press, 3rd edition, August 2014.

[4] M. H. Bhuyan, D. K. Bhattacharyya, and J. K. Kalita. Network anomaly detection: Methods, systems and tools. *IEEE Communications Surveys Tutorials*, 16(1):303–336, 2014.

[5] N. Brownlee, C. Mills, and G. Ruth. RFC 2722 - Traffic flow measurement: Architecture. Technical report, IETF, October 1999.

[6] S. Burschka, B. Dupasquier, A. Fiaux, and T. Rühl. Tranalyzer. `http://tranalyzer.com/`. Accessed: 2016-12-07.

[7] Cisco Systems. Cisco IOS Netflow. `www.cisco.com/c/en/us/products/ios-nx-os-software/ios-netflow/`. Accessed: 2017-02-15.

[8] Cisco Systems. Snort - Network intrusion detection and prevention system. `https://www.snort.org`. Accessed: 2016-12-07.

[9] Damballa. First Zeus, now SpyEye. Look at the source code now! `https://www.damballa.com/?p=8357`. Accessed: 2016-06-12.

[10] E. De la Hoz, E. De La Hoz, A. Ortiz, J. Ortega, and B. Prieto. PCA filtering and probabilistic SOM for network intrusion detection. *Neurocomputing*, 164:71–81, 2015.

[11] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38, 1977.

[12] O. J. Dunn. Multiple comparisons among means. *Journal of the American Statistical Association*, 56(293):52–64, 1961.

[13] ESET. First twitter-controlled android botnet discovered by ESET. `https://www.eset.com/a/ig1p02/`. Accessed: 2017-02-26.

[14] S. García, A. Fernández, J. Luengo, and F. Herrera. Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences*, 180(10):2044–2064, may 2010.

[15] S. García, M. Grill, J. Stiborek, and A. Zunino. An empirical comparison of botnet detection methods. *Computers & Security*, 45:100–123, 2014.

[16] C. T. Gimnez, A. P. Villegas, and G. Á. Marañón. HTTP data set CSIC 2010. `http://www.isi.csic.es/dataset`, 2010.

[17] D. Goodin. Record-breaking ddos reportedly delivered by >145k hacked cameras. `https://arstechnica.com/?post_type=post&p=966459`. Accessed: 2016-12-26.

[18] G. Gu, R. Perdisci, J. Zhang, and W. Lee. Botminer: clustering analysis of network traffic for protocol- and structure-independent botnet detection. In *Proceedings of the 17th USENIX Security Symposium*, pages 139–154, 2008.

[19] G. Gu, P. Porras, V. Yegneswaran, M. Fong, and W. Lee. BotHunter: Detecting malware infection through IDS-driven dialog correlation. In *Proceedings of the 16th USENIX Security Symposium*, 2007.

[20] F. Haddadi, D. C. Le, L. Porter, and A. N. Zincir-Heywood. On the effectiveness of different botnet detection approaches. In *Proceedings of 11th International Conference on Information Security Practice and Experience, ISPEC 2015*. Springer International Publishing, May 2015.

[21] F. Haddadi, D. Runkel, A. N. Zincir-Heywood, and M. I. Heywood. On botnet behaviour analysis using GP and C4.5. In *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pages 1253–1260, New York, NY, USA, 2014. ACM.

[22] F. Haddadi and A. N. Zincir-Heywood. Benchmarking the effect of flow exporters and protocol filters on botnet traffic classification. *IEEE Systems Journal*, 10:1390 – 1401, 2016.

[23] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explorations Newsletter*, 11(1):10–18, November 2009.

[24] B. Herzberg, D. Bekerman, and I. Zeifman. Breaking down Mirai: An IoT DDoS botnet analysis. `https://www.incapsula.com/blog/malware-analysis-mirai-ddos-botnet.html`. Accessed: 2017-02-26.

[25] J. L. Hodges and E. L. Lehmann. Rank methods for combination of independent experiments in analysis of variance. *The Annals of Mathematical Statistics*, 33(2):482–497, 1962.

[26] D. Ippoliti and X. Zhou. An adaptive growing hierarchical self organizing map for network intrusion detection. In *Proceedings of the 19th International Conference on Computer Communications and Networks*, pages 1–7, 2010.

[27] D. Ippoliti and X. Zhou. A-GHSOM: An adaptive growing hierarchical self organizing map for network anomaly detection. *Journal of Parallel and Distributed Computing*, 72(12):1576–1590, 2012.

[28] Juniper Networks. Juniper Flow Monitoring. `http://www.juniper.net/us/en/local/pdf/app-notes/3500204-en.pdf`. Accessed: 2017-02-15.

[29] Kaspersky lab. Kaspersky DDoS Intelligence Report for Q2 2016, August 2016.

[30] H. G. Kayacik, A. N. Zincir-Heywood, and M. I. Heywood. A hierarchical SOM-based intrusion detection system. *Engineering Applications of Artificial Intelligence*, 20(4):439–451, 2007.

[31] T. Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer Berlin Heidelberg, third edition, 2001.

[32] T. Kohonen. *MATLAB Implementations and Applications of the Self-Organizing Map*. Unigrafia Oy, Helsinki, Finland, 2014.

[33] Lawrence Berkeley National Laboratory and ICSI. LBNL enterprise trace repository. `http://www.icir.org/enterprise-tracing`, 2005.

[34] K. Leung and C. Leckie. Unsupervised anomaly detection in network intrusion detection using clusters. In *Proceedings of the Twenty-eighth Australasian conference on Computer Science*, volume 38, pages 333–342, 2005.

[35] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, Berkeley, Calif., 1967. University of California Press.

[36] M. Markou and S. Singh. Novelty detection: a review–part 1: statistical approaches. *Signal Processing*, 83(12):2481 – 2497, 2003.

[37] J. McHugh. Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by Lincoln Laboratory. *ACM Transactions on Information and System Security*, 3(4):262–294, 2000.

[38] S. Nagaraja, P. Mittal, C.-Y. Hong, M. Caesar, and N. Borisov. BotGrep: Finding P2P bots with structured graph analysis. In *Proceedings of the 19th USENIX Conference on Security*, Berkeley, CA, USA, 2010. USENIX Association.

[39] H. T. Nguyen, C. Torrano-Gimenez, G. Alvarez, S. Petrović, and K. Franke. Application of the generic feature selection measure in detection of web attacks. In *Proceedings of Computational Intelligence in Security for Information Systems: 4th International Conference, CISIS 2011*, pages 25–32. Springer Berlin Heidelberg, 2011.

[40] V. Paxson. The Bro network security monitor. `https://www.bro.org/`. Accessed: 2017-02-26.

[41] D. Pelleg and A. W. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 727–734, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[42] R. Perdisci, G. Gu, and W. Lee. Using an ensemble of one-class SVM classifiers to harden payload-based anomaly detection systems. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 488–498, Dec 2006.

[43] QoSient, LLC. Argus - Auditing Network Activity. `http://qosient.com/argus/`. Accessed: 2016-12-09.

[44] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[45] RSA Security LLC. Cybercrime 2015: An inside look at the changing threat landscape. Technical report, EMC, April 2015.

[46] B. Schneier. Botnets of things. `https://www.technologyreview.com/s/603500`. Accessed: 2017-02-26.

[47] S. S.C. Silva, R. M.P. Silva, R. C.G. Pinto, and R. M. Salles. Botnets: A survey. *Computer Networks*, 57(2):378 – 403, 2013.

[48] W. T. Strayer, D. Lapsely, R. Walsh, and C. Livadas. *Botnet Detection Based on Network Behavior*, pages 1–24. Springer US, Boston, MA, 2008.

[49] G. Szabó, D. Orincsay, S. Malomsoky, and I. Szabó. On the validation of traffic classification algorithms. In *Proceedings of the 9th International conference on Passive and active network measurement*, pages 72–81, 2008.

[50] The Honeynet Project. French Chapter. `http://www.honeynet.org/chapters/france`. Accessed: 2016-09-10.

[51] A. Ultsch and H. P. Siemon. Kohonen's self organizing feature maps for exploratory data analysis. In *Proceedings of the International Neural Network Conference (INNC-90)*, pages 305–308, July 1990.

[52] L. Van Der Maaten and G. Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[53] K. Veeramachaneni, I. Arnaldo, V. Korrapati, C. Bassias, and K. Li. AIˆ2: Training a big data machine to defend. In *2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity)*, pages 49–54. IEEE, April 2016.

[54] J. Vesanto, J. Himberg, E. Alhoniemi, and J. Parhankangas. SOM Toolbox for Matlab. Technical report, Laboratory of Information and Computer Science, Helsinki University of Technology, April 2000.

[55] P. Winter, E. Hermann, and M. Zeilinger. Inductive intrusion detection in flow-based network data using one-class support vector machines. In *Proceedings of the 4th IFIP International Conference on New Technologies, Mobility and Security*, pages 1–5, Feb 2011.

[56] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z.-H. Zhou, M. Steinbach, D. J. Hand, and D. Steinberg. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14(1):1–37, 2008.

[57] P. Wurzinger, L. Bilge, T. Holz, J. Goebel, C. Kruegel, and E. Kirda. Automatically generating models for botnet detection. In *Proceedings of the 14th European Conference on Research in Computer Security*, pages 232–249, 2009.

[58] Q. Yan, Y. Zheng, T. Jiang, W. Lou, and Y. T. Hou. PeerClean: Unveiling peer-to-peer botnets through dynamic group behavior analysis. In *2015 IEEE Conference on Computer Communications (INFOCOM)*, pages 316–324, April 2015.

[59] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani, and D. Garant. Botnet detection based on traffic behavior analysis and flow intervals. *Computers & Security*, 39:2–16, 2013.

Appendix A

Data Features

A.1 List of flow features exported using Tranalyzer

Tranalyzer can extract uni-directional flows with a wide range of features from network captures of live interface, based on the chosen plugins [6]. Designed for simplicity, performance and scalability, it generates statistics from key parameters of IPv4/IPv6 Tcpdump traces, which can be live captured from supported network interfaces, or packet capture files. A wide range of modules, or plugins are provided with Tranalyzer, each tailored to one or several fields of packet header at Link, Internet, Transport or even Application layer of the Internet protocol suite. The plugins enable network administrators not only to acquire interested information of flows in many categories, including time, inter-arrival, packet statistics (e.g. number of packets, number of bytes, bytes per packet, packet per second, ...), histogram, and flag information, but also to decode packet content and analyze payload when necessary. In this thesis, the seven most popular Tranalyzer plugins are employed, Table A.1. 71 numerical features, in which IP addresses and port numbers are not included, are then selected from the plugins' output for training and evaluating the systems in this thesis, Table A.2.

Table A.1: List of employed Tranalyzer plugins

Ind.	**Plugin**	**Description**
1	basicFlow	host identification fields and timing information
2	basicStats	basic layer four statistics for each flow
3	tcpFlags	contains IP and TCP header information encountered during the lifetime of a flow for troubleshooting purposes
4	icmpDecode	analyzes ICMP traffic and provides absolute and relative statistics
5	connStat	counts the connections between different IPs and ports per flow and during the pcap lifetime in order to produce an operational picture for anomaly detection
6	pktSIATHisto	records the packet length and inter-arrival time of a flow
7	descriptiveStats	calculates various statistics about a flow, using output from pktSIATHisto

Table A.2: List of used flow features from Tranalyzer

Ind.	Plugin	Flow feature	Description
1	1	Duration	Flow duration
2	1	ETHVlanID	VLAN number (inner VLAN)
3	2	numPktsSnt	Number of transmitted packets
4	2	numPktsRcvd	Number of received packets
5	2	numBytesSnt	Number of transmitted bytes
6	2	numBytesRcvd	Number of received bytes
7	2	minPktSz	Minimum layer 3 packet size
8	2	maxPktSz	Maximum layer 3 packet size
9	2	avePktSize	Average layer 3 packet size
10	2	pktps	Sent packets per second
11	2	bytps	Sent bytes per second
12	2	pktAsm	Packet stream asymmetry
13	2	bytAsm	Byte stream asymmetry
14	3	ipMindIPID	IP Minimum delta IP Identification
15	3	ipMaxdIPID	IP Maximum delta IP Identification
16	3	ipMinTTL	IP Minimum Time to Live (TTL)
17	3	ipMaxTTL	IP Maximum Time to Live (TTL)
18	3	ipTTLChg	IP TTL Change Count
19	3	ipOptCnt	IP options count
20	3	tcpPSeqCnt	TCP packet sequence count
21	3	tcpSeqSntBytes	TCP sent seq diff bytes
22	3	tcpSeqFaultCnt	TCP sequence number fault count
23	3	tcpPAckCnt	TCP packet acknowledgement (ACK) count
24	3	tcpFlwLssAckRcvdBytes	TCP flawless ack received bytes

ind.	Plugin	Flow feature	Description
25	3	tcpAckFaultCnt	TCP ack number fault count
26	3	tcpInitWinSz	TCP initial effective window size
27	3	tcpAveWinSz	TCP average effective window size
28	3	tcpMinWinSz	TCP minimum effective window size
29	3	tcpMaxWinSz	TCP maximum effective window size
30	3	tcpWinSzDwnCnt	TCP effective window size change down count
31	3	tcpWinSzUpCnt	TCP effective window size change up count
32	3	tcpWinSzChgDirCnt	TCP effective window size direction change count
33	3	tcpOptPktCnt	TCP options packet count
34	3	tcpOptCnt	TCP options count
35	3	tcpMSS	TCP Maximum Segment Length
36	3	tcpWS	TCP Window Scale
37	3	tcpSSASAATrip	(A) TCP Trip Time Syn, Syn-Ack; (B) TCP Trip Time Syn-Ack, Ack
38	3	tcpRTTSseqAA	(A) TCP Round Trip Time Syn, Syn-Ack, Ack; (B) TCP Round Trip Time Ack-Ack RTT
39	3	tcpRTTAckTripMin	TCP Ack Trip Minimum
40	3	tcpRTTAckTripMax	TCP Ack Trip Maximum
41	3	tcpRTTAckTripAve	TCP Ack Trip Average
42	4	icmpEchoSuccRatio	Echo reply/request success ratio
43	5	connSip	Number of connections from source IP to different hosts

ind.	Plugin	Flow feature	Description
44	5	connDip	Number of connections from destination IP to different hosts
45	5	connSipDip	Number of connections from source IP to destination IP
46	7	MinPl	Minimum packet length
47	7	MaxPl	Maximum packet length
48	7	MeanPl	Mean packet length
49	7	LowQuartilePl	Lower quartile of packet lengths
50	7	MedianPl	Median of packet lengths
51	7	UppQuartilePl	Upper quartile of packet lengths
52	7	IqdPl	Inter quartile distance of packet lengths
53	7	ModePl	Mode of packet lengths
54	7	RangePl	Range of packet lengths
55	7	StdPl	Standard deviation of packet lengths
56	7	RobStdPl	Robust standard deviation of packet lengths
57	7	SkewPl	Skewness of packet lengths
58	7	ExcPl	Excess of packet lengths
59	7	MinIat	Minimum inter-arrival time
60	7	MaxIat	Maximum inter-arrival time
61	7	MeanIat	Mean inter-arrival time
62	7	LowQuartileIat	Lower quartile of inter-arrival times
63	7	MedianIat	Median of inter-arrival times
64	7	UppQuartileIat	Upper quartile of inter-arrival times
65	7	IqdIat	Inter quartile distance of inter-arrival times
66	7	ModeIat	Mode of inter-arrival times

ind.	Plugin	Flow feature	Description
67	7	RangeIat	Range of inter-arrival times
68	7	StdIat	Standard deviation of inter-arrival times
69	7	RobStdIat	Robust standard deviation of inter-arrival times
70	7	SkewIat	Skewness of inter-arrival times
71	7	ExcIat	Excess of inter-arrival times

A.2 List of features used for web request log analysis

From the original web request provided by CSIC, a list of features are exported to represent each request by a numerical vector, Table A.3. A number of the features are taken from [39], while the rest are heuristically chosen.

Table A.3: List of features extracted from CSIC HTTP data set

Ind.	Type	Feature	Ind.	Type	Feature
1	real	request length	2	real	requested path's depth
3	real	total length of arguments	4	real	number of arguments
5	real	number of letters in arguments	6	real	number of digits in arguments
7	real	number of special characters in arguments	8	real	number of other characters in arguments
9	real	number of letters in path	10	real	number of digits in path
11	real	number of other characters in path	12	real	port number
13	real	payload length	14	binary	SQL in payload
15	binary	“login” fields in payload	16	binary	type img requested
17	binary	type js requested	19	binary	other type requested
19	binary	GET request	20	binary	POST request
21	binary	PUT request			

Appendix B

Chapter 4 Supplementary

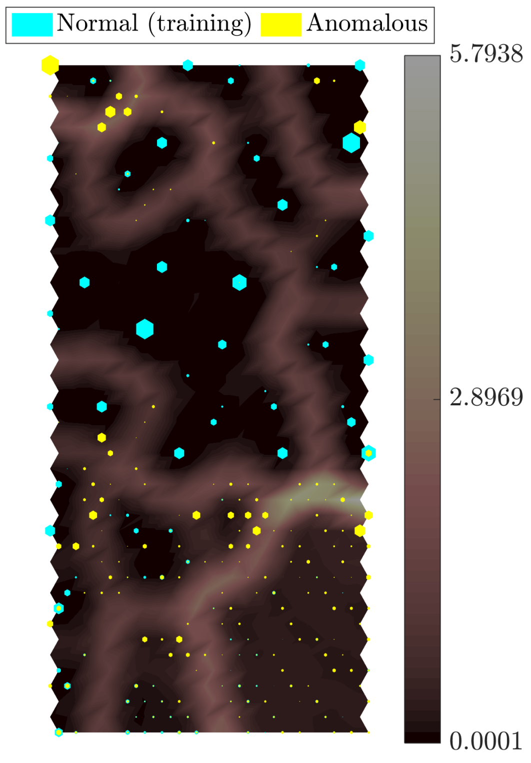
B.1 Training and Testing data separation

The training and testing data partition size of the data sets are presented in Table B.1. Basically the splitting procedure is based on stratified three-fold cross validation. However, for larger data sets - ISOT, CTU13a, c, h, i, j, m - each trial employs only the smaller partition of split data for training the SOM. Futhermore, since the HTTP-CSIC data set is provided with separate training and testing sets for normal web requests, the training - testing partitioning is only applied to the malicious web requests. From the full training partitions, which is used for training the SOMs in scheme (i), the training sets to be used in scheme (ii) and (iii) are obtained by retaining only Normal or only Malicious data, respectively. It should also be noted that the training and testing partitions used in Section 4.4.3 is listed as CTU13 in the table. For this set, only 50% the training data is randomly sampled to train the algorithms in each run.

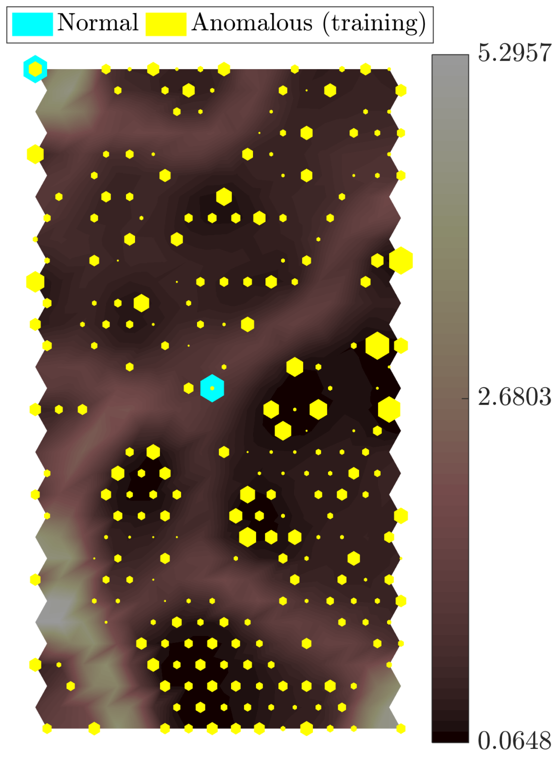
Table B.1: Training and testing partition sizes of the data sets

Data set	**Training partition size**			**Testing partition size**
	Scheme(i)	**Scheme(ii)**	**Scheme(iii)**	
CTU13a	23783	10129	13654	23840
CTU13b	20041	6080	13960	10020
CTU13c	47903	38963	8940	95808
CTU13d	18565	16845	1720	9283
CTU13e	3720	3119	601	1860
CTU13f	8083	4996	3087	4041
CTU13g	1160	1118	42	580
CTU13h	26317	24275	2042	52632
CTU13i	71652	9989	61663	143302
CTU13j	40733	5282	35451	81466
CTU13k	7255	1812	5443	3627
CTU13l	6531	5086	1445	3265
CTU13m	23981	10646	13335	47961
CTU13	203159			393696
ISOT	88294	70734	17560	176288
HTTP-CSIC	38323	28000	10323	33162

B.2 Hit Maps of Normal and malicious data on SOMs trained using schemes (ii) and (iii)

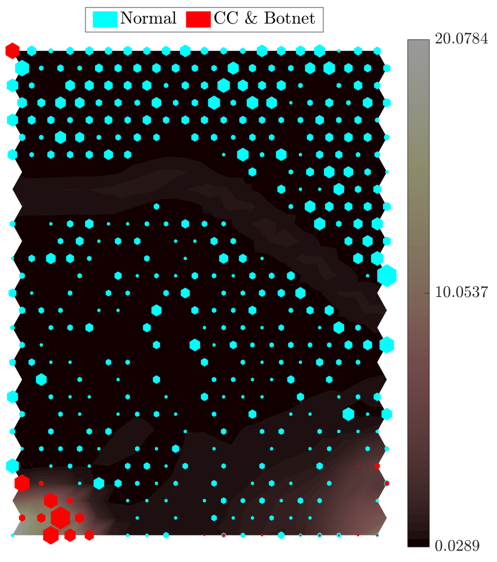


(a) SOM trained using Normal requests

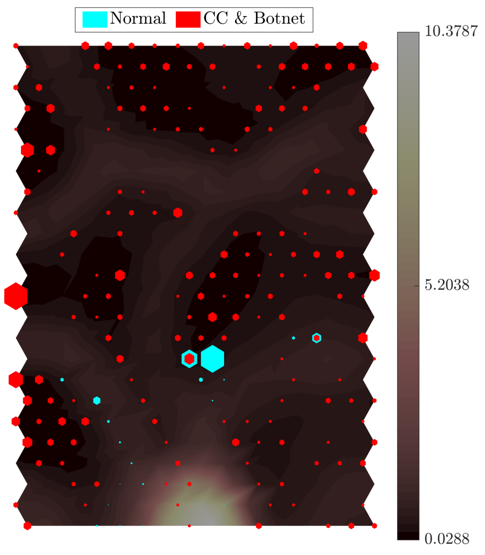


(b) SOM trained using malicious requests

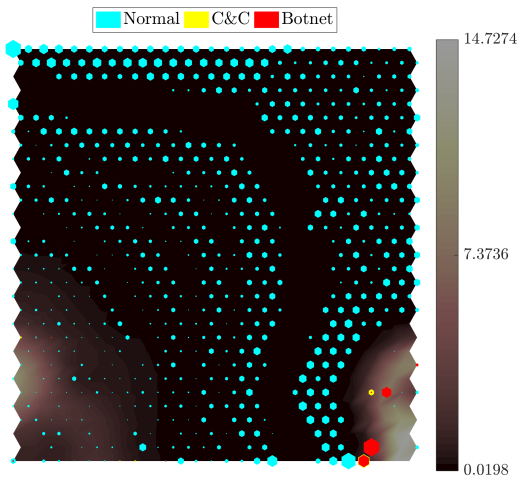
Figure B.1: Hit maps of HTTP-CSIC Normal and Malicious requests on SOMs trained using only Normal requests (left) and SOMs trained using only Malicious requests(right)



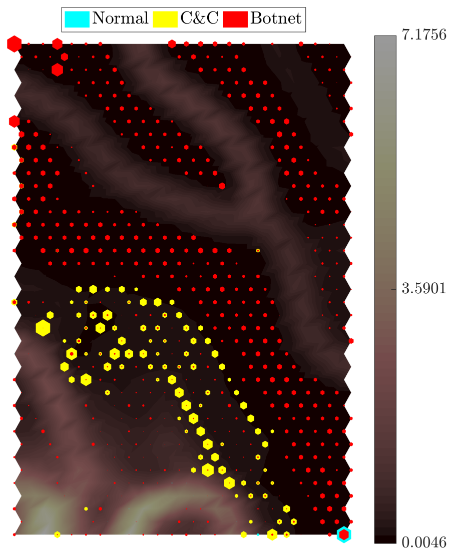
(a) SOM trained using Normal flows, CTU13l



(b) SOM trained using CC & Botnet flows, CTU13l

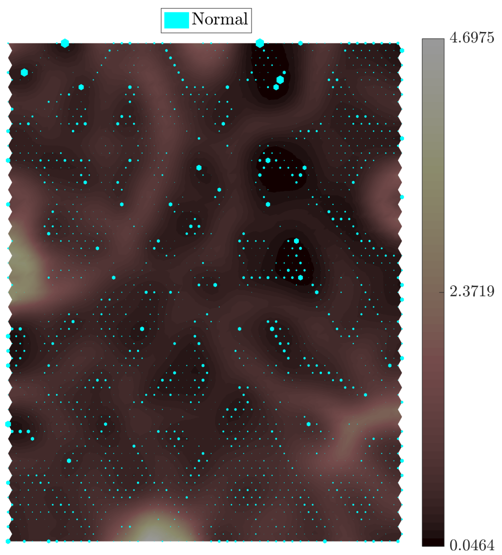


(c) SOM trained using Normal flows, CTU13m

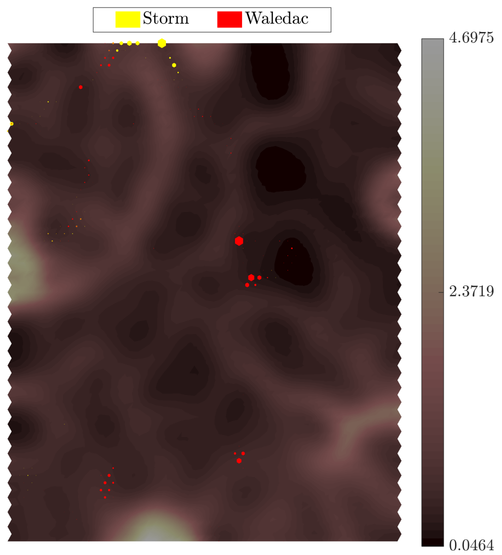


(d) SOM trained using CC & Botnet flows, CTU13m

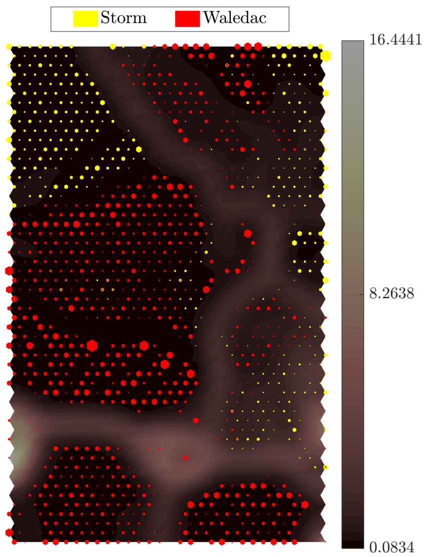
Figure B.2: Hit maps of CTU13 Normal, C&C, and Botnet flows on SOMs trained using only Normal flows (left) and SOMs trained using only CC & Botnet flows(right)



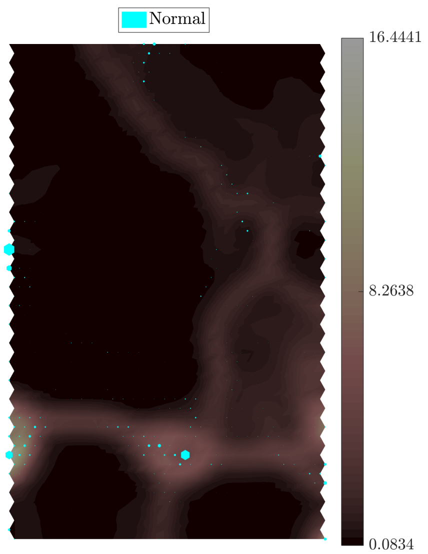
(a) Normal (training) flows on SOM trained using Normal flows



(b) Botnet flows on SOM trained using Normal flows



(c) Botnet (training) flows on SOM trained using Botnet flows



(d) Normal flows on SOM trained using Botnet flows

Figure B.3: Hit maps of ISOT Normal and Botnet flows on SOMs trained using only Normal flows (up) and SOMs trained using only CC & Botnet flows(down)

B.3 Detailed classification performances of SOM and other algorithms

Table B.2: Classification performance of SOM and other learning algorithms

Data set	Alg.	Accuracy	CDR	Detection rate			Precision
CTU13				**Normal**	**C&C**	**Botnet**	
CTU13a	SOM	98.13	97.44	99.23	95.77	97.33	99.44
	C4.5	99.83	92.97	99.98	79.05	99.88	99.99
	Naive Bayes	98.72	68.73	99.19	7.86	99.14	99.40
	X-means	98.04	98.73	99.74	99.71	96.75	99.81
	EM	77.57	85.87	98.84	97.28	61.49	98.88
CTU13b	SOM	96.61	95.68	99.37	92.17	95.51	99.73
	C4.5	99.60	96.29	99.93	89.13	99.79	99.97
	Naive Bayes	82.77	89.10	99.18	93.07	75.05	99.64
	X-means	80.75	90.21	99.81	99.26	71.55	99.92
	EM	81.61	89.62	97.32	97.53	74.01	98.82
CTU13c	SOM	99.67	99.78	99.60	99.95		98.30
	C4.5	99.94	99.89	99.97	99.80		99.87
	Naive Bayes	99.75	99.76	99.74	99.78		98.90
	X-Means	99.86	99.87	99.86	99.87		99.41
	EM	98.35	96.53	99.43	93.64		97.39
CTU13d	SOM	99.54	99.66	99.51	99.82		95.38
	C4.5	99.89	99.73	99.92	99.54		99.24
	Naive Bayes	98.62	97.92	98.78	97.05		89.05
	X-Means	99.84	99.91	99.83	99.98		98.35
	EM	89.37	92.44	88.67	96.21		46.43
CTU13e	SOM	98.83	99.21	98.64	99.78		93.40
	C4.5	99.94	99.90	99.96	99.84		99.79
	Naive Bayes	99.51	99.19	99.66	98.72		98.27
	X-Means	99.85	99.88	99.85	99.91		99.20
	EM	95.29	93.27	96.26	90.28		82.28

Data set	Alg.	Accuracy	CDR	Detection rate			Precision
CTU13f	SOM	99.24	99.37	98.82	99.92		98.13
	C4.5	99.90	99.90	99.90	99.91		99.83
	Naive Bayes	99.55	99.48	99.77	99.18		99.63
	X-Means	99.85	99.87	99.77	99.97		99.63
	EM	99.30	99.33	99.20	99.46		98.71
CTU13g	SOM	97.31	97.86	97.27	98.45		57.55
	C4.5	99.51	98.64	99.58	97.70		89.79
	Naive Bayes	99.74	98.22	99.86	96.59		96.28
	X-Means	99.74	98.60	99.83	97.38		95.49
	EM	88.19	81.58	88.71	74.44		19.85
CTU13h	SOM	99.56	99.48	99.57	99.39	99.47	95.48
	C4.5	99.94	99.45	99.98	98.80	99.56	99.78
	Naive Bayes	99.47	97.65	99.79	98.04	95.12	97.50
	X-means	99.45	99.22	99.53	100.00	98.13	94.69
	EM	55.94	82.22	52.61	99.21	94.83	14.84
CTU13i	SOM	96.42	97.34	99.18	96.90	95.95	99.87
	C4.5	99.67	95.54	99.71	87.06	99.87	99.95
	Naive Bayes	92.66	87.16	99.20	70.34	91.95	99.87
	X-means	89.90	95.85	99.75	99.69	88.11	99.96
	EM	81.45	87.02	96.03	86.06	78.96	99.28
CTU13j	SOM	99.81	99.45	98.97	99.94		99.85
	C4.5	99.96	99.88	99.78	99.98		99.97
	Naive Bayes	99.63	99.43	99.15	99.71		99.87
	X-Means	99.81	99.62	99.38	99.87		99.91
	EM	95.43	94.81	93.99	95.64		99.07
CTU13k	SOM	99.75	99.60	99.30	99.90		99.77
	C4.5	99.85	99.85	99.85	99.85		99.95
	Naive Bayes	98.84	99.20	99.92	98.48		99.97
	X-Means	99.90	99.89	99.87	99.90		99.96
	EM	92.49	90.79	87.37	94.20		95.73
CTU13l	SOM	99.14	99.35	98.83	99.87		97.32
	C4.5	99.74	99.57	99.87	99.26		99.54
	Naive Bayes	98.41	98.00	98.73	97.28		95.60
	X-Means	99.45	99.18	99.66	98.71		98.81
	EM	93.83	89.16	97.54	80.78		90.33

Data set	**Alg.**	**Accuracy**	**CDR**	**Detection rate**			**Precision**
CTU13m	SOM	96.59	93.22	99.43	85.74	94.50	99.55
	C4.5	99.76	93.77	99.99	81.50	99.81	99.99
	Naive Bayes	89.33	68.61	95.51	25.11	85.20	96.51
	X-means	70.98	79.36	99.91	90.91	47.26	99.93
	EM	65.93	62.38	58.92	56.47	71.75	72.80
				Normal	**Storm**	**Waledac**	
ISOT	SOM	95.29	93.60	95.91	91.18	93.70	85.70
	C4.5	99.25	97.66	99.87	95.87	97.24	99.49
	Naive Bayes	82.85	72.17	90.20	97.67	28.65	71.49
	X-means	87.23	83.64	88.69	80.27	81.96	67.74
	EM	34.40	42.79	30.30	42.42	55.66	23.73
				Normal	**Malicious**		
HTTP-CSIC	SOM	92.82	93.70	92.42	94.97		69.80
	C4.5	96.50	93.05	99.14	86.97		96.54
	Naive Bayes	84.08	72.61	92.86	52.35		66.96
	X-Means	74.93	83.51	68.37	98.65		46.31
	EM	74.86	74.97	74.78	75.16		45.18

B.3.1 Statistical test for analysis of the results

To provide statistical support for the analysis of the results, non-parametric tests are conducted due to the fact that the assumptions of parametric test can not be guaranteed. Specifically, Friedman Aligned-Ranks test [25] is employed to detect statistical significant differences among results of learning algorithms on the data sets, and the Bonferroni-Dunn post-hoc test [12] is used to identify the algorithms which actually differ from SOM. The tests are suggested to use in the field of machine learning when the number of algorithms for comparison is small [14].

Two tests are carried out with the ranking of Accuracy and Class-wise detection rate of the algorithms on the data sets as the test variables. The test results are summarized in Table B.3. With five algorithms and fifteen data sets, the aligned rank of the algorithms is ranging from 8 to 68, where the higher is the better. And, the statistic T is distributed according to the chi-square distribution with $5 - 1 = 4$ degrees of freedom. The aligned ranks of SOM (48.73 and 52.77) are better than that of Naive Bayes, X-means, and EM, and at a competitive level to the aligned ranks of C4.5 (54.80 and 54.40). After computing the ranks, the Friedman aligned-ranks test obtained statistic T and p-value of 34.92 and $4.84 \cdot 10^{-7}$ for accuracy, and 38.32 and $9.63 \cdot 10^{-8}$ for CDR, respectively. Consequently, the null-hypotheses stating that all algorithms perform equally in mean ranking of Accuracy and CDR are rejected, or the two tests detected significant differences between the algorithms.

Due to these differences, two post-hoc statistical analyses are required. The main algorithm of this thesis, SOM, is chosen as the control method to perform the post-hoc analyses for comparison with the rest of the algorithms. As it is shown in the table, there were significant differences between SOM and EM for both Accuracy and CDR (p-values of $5.83 \cdot 10^{-6}$ and $9.02 \cdot 10^{-7}$), and between SOM and Naive Bayes for CDR (p-value of 0.003). The post-hoc tests could not detect any significant differences between SOM, C4.5, and X-means in both Accuracy and CDR. The statistical tests support that the proposed approach in this thesis outperforms EM and Naive Bayes, and is very competitive when compared with C4.5 and X-means.

Table B.3: Statistical test results of performances of the five algorithms on the data sets. Post-hoc test shows the statistic z and Bonferroni adjusted p-value in parentheses, with SOM is the control algorithm

Algorithm	**Accuracy**			**Class-wise DR**		
	Average	**Rank**	**Post-hoc test**	**Average**	**Rank**	**Post-hoc test**
SOM	97.91	48.73		97.65	52.87	
C4.5	99.55	54.80	0.76 (1.00)	97.73	54.40	0.19 (1.00)
Naive Bayes	94.93	39.00	1.22 (0.88)	89.82	26.20	3.35 (0.003)
X-means	93.31	37.07	1.47 (0.57)	95.16	44.87	1.01 (1.00)
EM	81.60	10.40	4.82 ($5.83 \cdot 10^{-6}$)	84.18	11.67	5.18 ($9.02 \cdot 10^{-7}$)
Statistic T		34.92			38.32	
p-value		$4.84 \cdot 10^{-7}$			$9.63 \cdot 10^{-8}$	

B.4 Details of CTU13 Background data analysis

Table B.4: Distribution of CTU13 Background traffic flows on the trained SOMs

CTU13	**Training scheme (i)**			**Training scheme (ii)**		**Training scheme (iii)**	
	Normal	**Anomaly**	**Botnet**	**Normal**	**Anomaly**	**Not Botnet**	**Botnet**
a	53.00	44.67	2.32	51.13	48.87	74.18	25.82
b	58.74	40.33	0.93	60.11	39.89	26.17	73.83
c	57.67	38.19	4.14	54.39	45.61	87.85	12.15
d	67.75	29.64	2.60	69.91	30.09	66.14	33.86
e	65.25	30.14	4.61	74.33	25.67	15.52	84.48
f	65.81	33.91	0.28	67.24	32.76	68.88	31.12
g	63.76	29.80	6.45	55.61	44.39	37.44	62.56
h	65.30	31.59	3.11	62.67	37.33	52.83	47.17
i	65.71	32.70	1.59	56.80	43.20	87.88	12.12
j	60.94	32.90	6.17	61.29	38.71	36.55	63.45
k	52.63	40.94	6.43	33.55	66.45	56.32	43.68
l	56.17	24.86	18.97	56.37	43.63	43.69	56.31
m	54.02	41.39	4.59	49.76	50.24	33.34	66.66

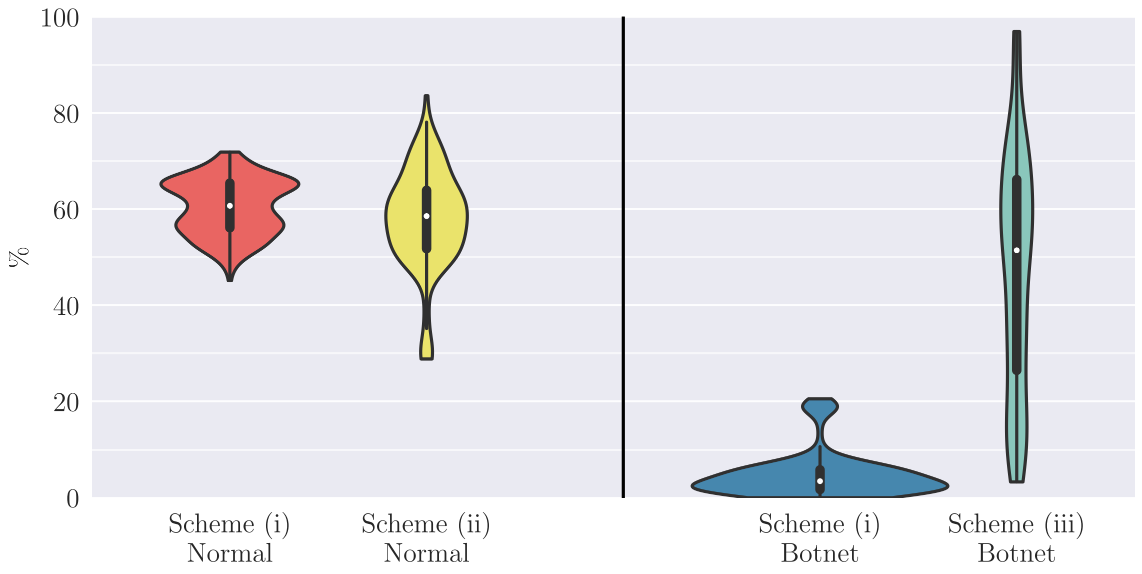


Figure B.4: Violin plots showing the percentages of Background traffic in CTU13 data sets labelled as Normal and Botnet by SOMs trained using the three schemes