Data Driven Countermeasures in Computer Networks

by

Ahmed Maher Mostafa

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

in

Software Engineering and Intelligent Systems

Department of Electrical and Computer Engineering

University of Alberta

Abstract

Network connectivity is an indispensable component of any computer related activities. Any computer-like machine is connected to some kind of a network, which is further connected to another computer network. Technological advances allow users to connect any devices and use multiple applications. As the result, the complexity of network systems is constantly growing.

The fact that almost all devices are connected means that any attempt to break into a device or a system is occurring via a computer network. In other words, networks and their connected devices are targets of intrusions. The increased dependency on computer systems and the growing number of malicious activities increase a pressure on governmental, industrial and private institutions to utilize security systems capable of monitoring and analyzing network traffic, and detecting malicious or suspicious activities.

The research topic of Intrusion Detection becomes of special importance. Intrusion Detection involves two processes: detecting cyber attacks using well-know attack patterns – Signature Detection (Signature ID), or identifying anomalous behaviour of network traffic – Anomaly Detection (Anomaly ID). Recently, a hybrid approach has emerged that tries to harvest advantages of both signature and anomaly detection methods.

Despite the promising prospects, hybrid Intrusion Detection System (IDS)s still need to demonstrate their usefulness and good performance. Some

of problems and unsolved issues are related to the lack of: 1) a detailed representation of network traffic that allows to recognize small differences in a traffic or detect non-standard cyber attacks; and 2) an accurately labeled data that contain different cases of anomalies and attacks.

In this thesis, we propose a *Two-stage Hybrid Intrusion Detection System* able to detect anomalies and attacks, and reporting its findings to a security administrator. The system includes three sub-systems: 1) *Network Data Collecting and Processing (NetDataCoP)* Module; 2) *Anomaly Detection* Module; and 3) *Signature Detection* Module.

In order to design this system, a novel, comprehensive and multi-perspective description of network traffic has been proposed. It includes more than hundred features representing a traffic at different levels of granularity, at different layers, involving different protocols, all determined over different temporal intervals. One of essential aspects of the system is its ability to identify and categorize application connections that utilize UDP – a connection-less protocol. The processes of detecting anomalies and attacks are built using algorithms of Machine Learning. The system utilize elements of Evidence Theory not only to detect anomaly/attack but also to identify a degree of confidence in its detection outcomes.

The results are highly promising. The implemented and tested system provides a very good performance, i.e., a number of false negatives (assuming anomaly/attack as a positive event) is zero, with a minimal number of false positives at the same time.

I would like to dedicate this work to my family.

Acknowledgments

Thanks GOD.

I want to take this opportunity to thank all people that had an influence in the work presented in this thesis. First and foremost, I would like to thank my supervisor, Dr.Marek Reformat for his time, inspiration, insightful comments and suggestions about my work. Also, I would like to extend my thanks to the University of Alberta and all its staff and colleagues who guided me in achieving this thesis.

I am also grateful to Electrical and Computer Engineering Department's Staff for generously sharing their intelligent ideas and comments that were valuable sources of encouragement.

Table of Contents

List of Figures

List of Tables

Abbreviations

DARPA	Defense Advanced Research Projects Agency
DOS	Denial of Service
EM	Expectation Maximization clustering
HIDS	Host IDS
ID	Intrusion Detection
IDS	Intrusion Detection System
IPS	Intrusion Prevention System
LR	Logistic Regression
NB	Naïve Bayes
NIDS	Network IDS
PROBE	Probe
R2L	Remote to Local
SVM	Support Vector Machine
TBM	Transferable Belief Model
TCP/IP	Transmission Control Protocol/Internet Protocol
U2R	User to Root

Chapter 1

Introduction

The increased dependency of multiple organizations – from governmental to business and private – on computer systems and networks has made *cyber security* an important issue. Governments, organizations, companies, and military rely on computer systems and computer networks to fulfill their daily operations. Presently, almost every computer-like device is connected to some kind of a computer network that is further connected to another computer network. Thus, connectivity is an indispensable part of any computer related activities. At the same time, complexity of network systems is growing – more devices are connected, more applications are used. The fact that almost all devices are connected means that any attempt to break into a device or a system is occurring via a computer network. Whether computer networks are private or public, they are always subject to espionage and infiltration on the pretence of theft, denial of availability, tampering, or destruction. Therefore, defensive procedures and security mechanisms should be implemented to preserve the integrity and to ensure uninterrupted use of these networks. In the last decade, technological advancements have grown exponentially. Security violations take place more often and cyber defence and security become of increased importance. Intrusion Detection is one of the most important topics of

cyber security. Intrusion Detection Systems (IDSs) monitor computer networks and/or software systems in order to detect malicious\suspicious activities *i.e.*, intrusions. Intrusions can be either attacks or anomalies. Therefore, IDSs use two fundamental methodologies, signature intrusion detection to detect attacks and anomaly intrusion detection to detect anomalies.

Signature Intrusion Detection, also known as misuse detection, searches for traces of attacks within network traffic. Signature Intrusion Detection System utilizes attack signatures as a source of knowledge to detect attacks. Attack signatures consist of a set of features and their respective values. Signatures are designed by security experts based on knowledge about past intrusions and vulnerabilities.

Signature Intrusion Detection (ID) has the advantage of detecting attacks with high accuracy (**high true negative rate**), also, they have the ability to provide detailed information about detected attacks. This information include the attack source(s), attack targets, used protocol, exchanged packets, elapsed attack duration, attack instances, and established connections. On the contrary, signature ID has the drawback of unable to detect new or unknown attacks (**high false positive rate**), besides, it has the limitation of continuously keeping the signatures up-to-date.

On the other hand, anomaly ID detects violations or deviations from a pre-defined normal threshold(s). Anomaly IDS utilizes anomaly profiles that models normal behavior in the form of thresholds. Anomaly profiles also consist of a set of features and their respective values; profiles are designed based on knowledge about past patterns of normal behavior. Anomaly ID has the advantage of detecting new '*zero day*' attacks as anomalies (**high true positive rate**). Though, anomalies can indicate attack attempt or new attack pattern, but the detected anomalies might also be new legitimate normal behavior (**high false**

negative rate), in other words, abnormal behavior is not always an indication of intrusion.

Anomaly ID has the advantage of providing detailed information about the detected anomaly, information that include the source of anomaly, the violated protocol, and the violated TCP/IP layer. The advantage gives rise to the disadvantage, anomaly ID lacks to provide detailed information about the type and nature of detected attacks.

In order to pursue the already stated research aspects we must have network traffic data. Intrusion Detection can be implemented to detect online intrusions within real-time captured network data, or it can be implemented to detect intrusions within off-line pre-stored data. Nonetheless, the application of machine learning and computational intelligence techniques on real-time network traffic requires the use of pre-stored data, to train classifiers and build models during the training phase. This task is a rather difficult, if not impossible.

1.1 Motivations

Our work targets research areas related to application of computational intelligence and machine learning techniques to enhance intrusion detection technologies. We focus on designing and developing intrusion detection algorithms using data-centric methods and techniques suitable for detecting and identifying malicious activities targeting whole sub-networks or individual resources.

The main motivation is to enhance capabilities of IDSs so they are able not only to detect an intrusion, but also to estimate indicators of potential intrusive activities in a qualitative and quantitative way. Those capabilities can be achieved by applying a combination of computational intelligence algorithms and machine

learning techniques to online and/or offline processing and analyzing of network traffic data.

1.2 Objective

The increased number of cyber attacks and intrusions requires more comprehensive methods and approaches used to develop effective IDSs. Machine learning and computational intelligence have been applied to enhance effectiveness and efficiency of these systems.

Researchers investigate different methods to construct attack models leading to signature ID; or normal network traffic models leading to anomaly ID. These models predict the status of network traffic, thus the intrusion detection problem is transformed into a classification problem. In addition, researchers also try to combine both methodologies, i.e., signature and anomaly ID, to construct *hybrid data-driven intelligent intrusion detection systems*. These hybrid detection systems aim to combine advantages of both methodologies, and at the same time decrease their limitations. A research proposed in this thesis aims at developing a methodology for building hybrid network Intrusion Detection System for monitoring activities of a computer system, analyzing data collected during that monitoring, and identifying potential cyber intrusions, i.e., attacks and anomalies. The system is steadfast in its implementation:

- It performs detailed byte-wise interpretation of raw network traffic to infer the fields of protocols, and conversely interpret them.
- It identifies and reconstructs Internet (ICMP) and Transport (TCP and UDP) connections.
- It investigates traffic using a temporal view of it, i.e., it continuously

analyzes a network traffic data collected over define time intervals, unlike other systems that utilize packets or connections.

- It performs a continuous anomaly intrusion detection via observing network traffic at *two* different levels of granularity: network-level and host-level; such an approach reinforces the detection capabilities.

- It carries out a continuous signature intrusion detection to ascertain attacks based on detecting attack signatures together with analysis of traffic on a network-level as well as a host-level.

1.3 Contributions

The performed literature review indicates that research in Intrusion Detection involves multiple topics that are investigated individually or together. It seems that the most significant topics are:

Intrusion Detection Datasets: development of Intrusion Detection Systems is very dependent on availability of traffic data; good quality datasets which describe network traffic and contain information about normal and anomalous behavior of a network, as well as different types of attack are essential; a lot of activities is dedicated to generation of data and building suitable datasets.

Anomaly Intrusion Detection: anomaly means any kind of behavior in a network which is not normal; a process of Anomaly Intrusion Detection allows to detect anomalous (not normal) behavior; multiple Machine Learning (ML) and Artificial Intelligence techniques are used to classify anomalous network traffic.

Signature Intrusion Detection: it is a process of detecting specific attacks; and there are two approaches used to detect attacks: 1) an academic approach focusing on development of classifiers able to distinguish a given type of attack; and 2) a commercial approach targeting development of hardware and/or software solutions that compare network traffic against the stored attack signatures.

Versatility of research topics in the area of intrusion deception combined with multiple goals and objectives have lead to a number of contributions. They 'touch' all three areas of intrusion detection research presented above. In a nutshell, the contributions are following.

1. **Multi-perspective Description of Network Traffic:** We performed a through analysis of different network traffic and attacks. As the result, we identified a comprehensive set of features that thoroughly describes any type of network traffic.

2. **Improved Attack Signatures:** The analysis of attacks and the proposed network traffic description allowed us to update attack signatures. The updates include: 1) new/improved values of features of attack signatures tailored, on some occasions, using specific network topology; and 2) addition of new features for signatures of different attacks.

3. **Network Data Collecting and Processing System (NetDataCoP):** We designed and implemented a system called NetDataCoP. It has the capability to collect and interpret network traffic data using **three** different points of view: data packet-wise, connection-wise, and temporal-wise. The system is able to: 1) capture a byte-wise network traffic, process it, and change it into data packets; 2) identify and reconstruct logical connections

at different layers of a network stack; and 3) process network traffic, i.e., data packets and connections, and construct time interval datasets.

4. **Identification and Reconstruction of Logical Connections for UDP-based Applications:** We designed and implemented a novel methodology for identification and reconstruction of logical connections of UDP-based applications. We use Deep Packet Inspection (DPI) to understand the behavior of UDP-based applications. We propose an algorithm for detecting ICMP and TCP connections. Its implementation constitutes *Connection Identification and Reconstruction Module* that is a part of NetDataCoP.

5. **Network Anomaly Profile:** The multi-perspective description of network traffic together with an algorithm for reconstruction of connections have led us to proposing a threshold-based network anomaly profile. The novelty of our network anomaly profile comes from application of a comprehensive set of features, which enable capturing multiple aspects of a network traffic simultaneously.

6. **Evidence-based Anomaly Intrusion Detection System:** We propose a methodology for building a system able to detect anomalies using as its base analysis of network traffic represented as a continuous stream of data. A temporal-wise processing of network traffic is performed using a specified time interval (one minute in our case). The analyzed data is an input to several classifiers built using network-level and host-level data to distinguish between normal and anomalous traffic. Elements of *Evidence Theory* are used to enhance the detection capabilities by combining the results of classifiers and provide a probabilistic outcome about the state of a network traffic.

7. **Evidence-based Signature Intrusion Detection System:** We construct a number of classifiers representing different attacks on network as well as host levels, and utilize them to distinguish between several attacks. Additionally, we develop and implement procedures for further analysis of network traffic data that leads to obtaining a 'full' information about detected type of attack. These procedures are built based on our updated attack signatures.

8. **Two-stage Hybrid Intrusion Detection System:** The proposed anomaly and signature intrusion detection systems are integrated into one system named *two-stage hybrid intrusion detection system*. First, the anomaly detection process is applied to the time-wise processed network traffic to assess a degree of network anomalousness. Once it is identified to be anomalous, the signature detection system is activated to investigate the traffic data for possible attacks.

9. **Intrusion Detection Benchmark Datasets:** We have created an Intrusion Detection benchmark dataset. It consists of data points representing network traffic aggregated over time intervals of one minute. The dataset is fully formatted and all data points are labeled, and available for public use by researchers.

1.4 Thesis Outline

The rest of the thesis is organized as follows: Chapter 2 provides a comprehensive background about Intrusion Detection, Machine Learning classifiers, and Evidence Theory, along with some of the latest related research studies. Chapter 3 introduces the core of our work, it starts with network traffic

classification (Section 3.1), followed by the analysis of canonical and updated attack signatures for our selected set of attacks (Section 3.2). It continues to analyze well known intrusion detection datasets (Section 3.3), and explains our proposed multi-perspective feature-based network traffic description (Section 3.4), our proposed network anomaly profile (Section 3.5), and concludes with a detailed analysis of our proposed network-tailored attack signatures (Section 3.6).

Chapter 4 demonstrates the overall proposal of our system. Chapter 5 presents a thorough explanation of NetDataCoP module and its included modules. Afterwards, Chapter 6 handles the process of generating network traffic, concerning the structure of our testbed network (Section 6.1), the generated network traffic data (Section 6.2), and our proposed ID benchmark dataset (Section 6.3).

Chapter 7 manifests our proposed Anomaly ID stage, with detailed justification and description of its included sub-modules; Global Network-level Machine Learning-based Anomaly Detection (Section 7.1), and Local Host-level Threshold-based Anomaly Detection (Section 7.2). Likewise, Chapter 8 illustrates our proposed Signature ID stage, with thorough explanation of its included sub-modules; Global Network-level Machine Learning-based Signature Detection (Section 8.1), and Attack Detection Modules (Section 8.2).

Chapter 9 shows detailed hybrid intrusion detection case studies. The utilization is demonstrated by using 3 network traffic minutes; one normal, and 2 attack minutes.

Chapter 10 concludes the thesis with a summary of contributions and results of work, along with promising future research points.

Appendix A to Appendix D provide extra details about; our proposed network anomaly profile, the identification and reconstruction process of logical

connections for UDP-based applications, the features of our proposed multi-perspective network traffic description, and finally the deciphered TCP/IP protocols.

Chapter 2

Background and Related Work

In this section, we briefly present some aspects of background relevant to distributed environment, cyber security, as well as data mining and computational intelligence (Section 2.1). We focus on intrusion detection, especially attack types and their severity levels, and main methodologies used for intrusion detection. We also provide some examples of research work in our scope of study (Section 2.2).

2.1 Background

An IDS has to monitor not only local internal network traffic but also incoming traffic from external networks. An individual – internal or external – activity can be legitimate or intrusive. Given the intelligence of attackers, they rarely execute explicit intrusive activities, but they try to make their intrusions seem legitimate by hiding them within normal activities. Also, an activity can be individually legitimate, but when it is seen within a certain context then it is deemed intrusive (*i.e.*, Denial of Service attacks). Having explained all this, then the task of an IDS monitoring internal and external network traffic, examining the legitimacy of events and activities, becomes a hard task which needs intense and accurate

research. The use of Machine Learning and Computational Intelligence (CI) techniques helps in elevating the capabilities of IDS to perform its task in a data-driven intelligent manner.

The next 4 sections are used to explain multiple aspects: Intrusion Detection (next section), Machine Learning classifiers (Section 2.1.2), and elements of evidence theory (Sections 2.1.3 and 2.1.4).

2.1.1 Intrusion Detection

Intrusion Detection is the process of monitoring a network or a set of computers for signs of incidents, whether these incidents are violations that constitute an attack or indications of an imminent attack [10][11][12]

2.1.1.1 Intrusion Detection System (IDS) and Intrusion Prevention System (IPS)

An IDS is a device or software that automates the process of detecting intrusions. The main task of IDS is to detect attack patterns (signatures) within network traffic, and it can also identify reconnaissance or probe activity that precedes an attack. Additionally, IDS detects violations of security policies or deviations from acceptable security practices (anomalies). Finally, IDS contains a notification module that notifies security administrators of the detected attacks or identified violations. The notification contains information that IDS has logged about the detected incident(s), which can be displayed in the form of a message sent to a monitoring terminal or in the form of a report sent to the security administrator [10][11][12].

An Intrusion Prevention System (IPS) is an IDS with additional prevention capabilities. These capabilities are used to prevent, slow, or cripple ongoing

detected incidents. Security administrators can use IPS as IDS by switching off the prevention features [11].

2.1.1.2 Intrusion Detection Methodologies

IDS can work using several detection methodologies, whether they are separated or integrated. Based on the detection methodology, IDS can be classified into signature and anomaly IDS [10][11][12]. In this situation, integrating the two methodologies in one detection engine produces a hybrid IDS.

Signature is a known pattern that identifies an attack. Signature (or misuse) detection is the process in which detected incidents are compared against attack signatures to determine if the current incident represents an attack [10][11][12]. A Signature IDS works like anti-virus servers. An anti-virus server can only detect viruses whose definitions exist in the server's virus database. Likewise, the signature IDS can only detect those attacks whose signatures exist in the IDS database.

Anomaly detection is the process of comparing the detected incidents against a pre-constructed anomaly profile that represents normal behavior. Profiles can be created for several entities; users, computers, connections, protocols, layers or even an entire network. Anomaly detection is very useful in detecting new or unknown (zero-day) attacks [10][11][12]. Also, anomaly profiles are created from data captured over a time interval that could range from a day to an entire year. This duration can be considered as the training period of the anomaly-based IDS [10][11]. Furthermore, an anomaly profile can be static or dynamic. When static anomaly profiles are created they do not change unless the security administrator instructs the IDS to do so. Dynamic anomaly profiles have the ability to periodically change to adapt to the new normal behavioral aspects of the entity. Unfortunately, this type of profiles is susceptible to intrusions.

An attacker who knows that the anomaly profile is dynamic can perform small incremental anomalous incidents in order to shape and push the thresholds of the profile to a limit that allows him to perform attacks without being detected [11].

2.1.1.3 Intrusion Detection Technologies

Alternatively, based on IDS deployment (i.e., location) it can be classified into Network IDS (NIDS) and Host IDS (HIDS) [10][11][12]. A NIDS monitors and analyzes network traffic for malicious activities. Therefore, IDS should be placed at vital strategic point that allows it to have access to all incoming and outgoing traffic of all connected machines within the network. The literature in [12] suggests classification for NIDS into two types; online and offline NIDS. Online NIDS handles real time network traffic, while offline NIDS processes previously stored data. Also, the literature in [5] argues two classifications for NIDS. The first classification is format-based; hardware and software NIDS. The second classification is mode-based; inline and passive NIDS.

A HIDS monitors and analyzes internal host activities, whether they relate to its network traffic (inbound and outbound), system logs, processes, activity, and configuration [10][11][12]. This is usually done by embedding agents either on individual hosts or on nearby appliance. The main task of these agents is to monitor one or more hosts [11].

2.1.1.4 Intrusion Detection Datasets

Many intrusion detection researchers have introduced multiple intrusion detection datasets, which constitute a valid foundation for the implementation

and evaluation of intrusion detection approaches. The main purpose of these datasets is to provide network traffic data to researchers such they represent either known (attacks) and/or unknown (anomalies) intrusions.

In this section, we will highlight some of these intrusion detection datasets. There are *two* commonly used intrusion detection datasets: KDD CUP '99 and NSL-KDD; like any other intrusion detection research we were inspired by these datasets and their structure in our work.

KDD CUP '99 [10][13][14] is an intrusion detection dataset generated from the results of practical experiments conducted in Massachusetts Institute of Technology (MIT), and analyzed as intrusion detection project under the supervision of Defense Advanced Research Projects Agency (DARPA).

NSL-KDD [15][16] is an improved version of the KDD CUP '99 dataset; in which the deficiencies found in KDD CUP '99 were treated. The dataset was constructed in Network Security Lab at University of New Brunswick.

TUIDS [10][17] is a network intrusion detection dataset generated and produced at Tezpur University in India. The experiments were conducted on a smaller scale network environment than that of KDD CUP '99. Also, it simulates a different group of attacks than that within KDD CUP '99. The dataset includes two versions; packet-based and flow-based version with additional features.

2.1.1.5 Cyber Attacks and Taxonomy

The main focus of cyber security experts is on cyber attacks. Cyber attacks intend to cause devastating impacts; *e.g.*, theft, denial of availability, tampering, destruction, damage, or unauthorized access, according to the attacker's intension (infiltration or espionage). Cyber attacks are classified into categories based on some form of coherent relationship among groups of attacks. This

relationship is usually related to the execution pattern of the attack. Attacks leave multiple traces at network or host levels. Moreover, they can target many components of a system, from certain services on a target machine to variety of network resources.

2.1.1.5.1 The Taxonomy of Attacks The hierarchy of attack categories suggested by DARPA in its intrusion detection evaluation project (KDD CUP '99) [13][14][18] is used here due to its clarity and direct relation to attack execution pattern. According to [10][13][14][18], attacks are divided into four main categories; Denial of Service (DOS), Probe (Probe), Remote to Local (R2L), and User to Root (U2R). The name of each category signifies the behavior of its included attacks. DOS attacks are characterized by their ability to deny the legitimate use of one or more assets of the target machine. Denial of asset usage can be achieved by abusing protocol features, confusing TCP/IP stack of target machine, or by manipulating pre-existing unhandled bugs [10][13][14][18].

Probe attacks can be regarded as a staging phase before an actual damaging attack, in which scan and reconnaissance activity take place. Any attacker needs to have some information about the network or machine he intends to attack. Probe attacks provide the attacker with such information [10][13][14][18].

R2L attacks are used to gain local access to a remote target machine. The attacker does not have an account on the remote machine, so he tries to create or gain unauthorized access to a local account by exploiting vulnerabilities [10][13][14][18].

U2R attacks are similar to R2L attacks in the manner of exploiting vulnerabilities, but in this case the attacker exploits other types of vulnerabilities on the target machine to elevate his standard local account -which he might have got using R2L attacks- to a privileged (or administrator) root account [10][13][14][18]. In

our research, we have selected DOS and Probe attack categories.

DOS and Probe attack categories adopt a flow-based behavior while R2L and U2R attack categories appropriate a a packet-based behavior. The concept of packet-based behavior is derived from packet fields that hold specific values. These values can be used – as is – to identify attack features. Also, this behavior is based on a very few number of packets that hold these specific values. Therefore, attacks that follow a packet-based behavior are easy to detect and trivial to investigate. On the other hand, flow-based behavior depends not only on packet values, but also on the contextual pattern formed by multiple groups of packets. Their corresponding attack signatures are formed by extracting and processing several protocol fields to produce flow-based features. The investigation of flow-based feature values identifies a flow-based behavior attacks. As a result, contextually, flow-based behavior attacks can be mistaken as normal, thus they are hard to detect and valuable to study.

2.1.1.5.2 Cyber Attacks We chose seven attacks as selected set of attacks to investigate in our research; four DOS attacks and three Probe attacks. In this section we are going to present their basic definition and we present their detailed analysis later (Next Chapter). For DOS attack category we concentrate on; Local Area Network Denial, ICMP Flood, Smurf, and IGMP Flood. For Probe attack category, we selected; IPSweep, InsideSniffer, and PortScan.

LAND attack is characterized by sending a malformed packet having the same source and destination addresses to target machine. This is considered vulnerability in TCP/IP implementation. According to [13][14][18], the target machine become confused and locks up until manual reboot is performed. ICMP Flood [10] is identified by sending a burst of ICMP request packets in order to flood the target machine, to either consume his connection queue or consume

his CPU cycles.

Smurf attack [10][13][14][18] is similar to ICMP Flood, but instead of flooding the target machine with a burst of ICMP request packets, it is flooded with a burst of ICMP response packets. A malformed ICMP request packet is sent to the broadcast address xxx.xxx.xxx.255 by the attacker whose source address is the target's address. Consequently, all network devices respond back with ICMP response packets, thus flooding the target machine. Some variations of this attack are performed without the initial malformed packet. Again, Smurf starves the target machine's connection queue or CPU cycles. Though IGMP Flood might seem trivial, because the legitimate use of IGMP protocol is multicast, but it can be used by an attacker in a unicast manner to flood the target machine and try to exhaust its CPU cycles. In IGMP Flood, the attacker floods the target with bursts of IGMP packets.

IPSweep [10][13][14][18] sends many ICMP ping request packets to every possible IP address in the subnet to see which devices respond. In other words, it aims to scan the IP address range. InsideSniffer [10][13][14][18] attack tries to scan the names of the devices connected to a network. The attacker sniffs the network and finds which IP addresses send packets. Once a packet is found, its IP address is resolved to its respective name. Some new versions of the attack resolve an IP address range to their corresponding names without the initial sniffing. Finally, PortScan [10][13][14][18] attack tries to scan the ports of the target machine in an effort to find which ports are open or closed.

2.1.1.6 Anomalies and Anomaly Profiles

In this section we highlight anomalies from several aspects. First we explain the types of anomalies then we illustrate their application domains. There are mainly two types of anomalies; simple and complex, Figure 2.1 below. Simple anomaly

includes data whose value individually forms an outlier with respect to the rest of data. Complex anomaly does not entirely depend on the values, but depends on the context in which the data exits. i.e., the data can be anomalous in one context and normal in another context [19].

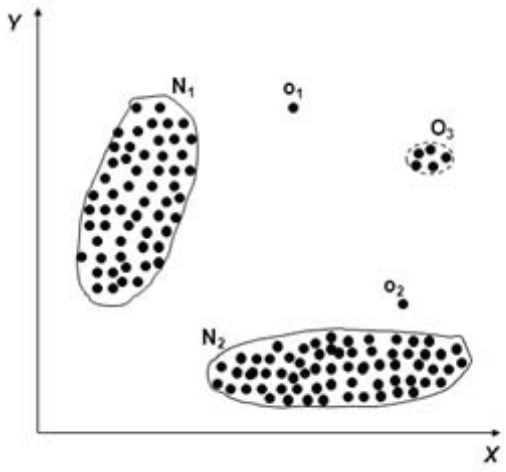


Figure 2.1: Anomalies.

The detection of anomalies has many application domains, among which intrusion detection, fraud detection, medical and healthcare anomaly detection, industrial damage detection, image processing, text anomaly detection, and sensor networks [19]. According to the application domain and the included entities within that domain, anomaly profiles for entities can be constructed. Profiles are utilized to represent normal behavior of entities in normal working conditions. In our case, we implement a network anomaly profile.

2.1.2 Machine Learning Techniques

Machine learning techniques can be categorized into either supervised or unsupervised techniques. The application of any machine learning technique implies a two-step procedure; inference of classification model, and application of inferred model upon new data (*i.e.*, classification). In the framework of our thesis we use four machine learning techniques; K-Means Clustering (K-Means), Decision Tree (DT), Logistic Regression (LR), and SVM; we are going to explain them in this section.

2.1.2.1 Decision Tree

Decision Tree is a supervised machine learning technique that has the ability to infer knowledge from data and represent it in the form of a model. Basically, the model is a decision tree consisting of several decision rules. Decision Trees consist of a group of nodes; they start with one node at the top called root node, and end with several nodes called leaf nodes. Each tree node contains a threshold value and a comparison sign, aside from the root node, each node is labeled with the learned class. Each path from root node to particular leaf node is considered a decision rule; decision rules can be regarded as a set of nested if-then statements [20].

Decision Tree learning algorithm has an important character; exclusivity. Consequently, exclusivity presents one advantage and two disadvantages: distinctive rules, replicated subtree, and model complexity. Decision Tree classifier advantage is the production of distinct non-conflicting non-overlapping rules, but sometimes this situation can generate complex models containing many rules. Also, the disadvantage of *replicated subtrees* problem, this happens when identical subtrees are learned from different parts

of the data which have similar knowledge [20].

Inference of classification model DT learning algorithm iteratively searches through the data to find an attribute whose value uniquely splits the data according to the given class labels. Once such attribute and value are found, they are treated as a root node. The algorithm recursively applies the same approach to each data split until no further splitting can be made. The difference between root node induction and other nodes is that each induced node is labeled with the respective class label [20].

Data classification Once a decision tree model is inferred, the algorithm uses that model to classify incoming new data. The attribute values of the new data are compared against the nodes of the decision tree by traversing through each decision rule, starting with the root node and ending with a leaf node. The result from traversing the decision tree is a class label that ties the given data to the predicted class [21].

2.1.2.2 Logistic Regression

Logistic Regression is a supervised machine learning technique that discovers knowledge from give data in the form of a logistic function. Logistic Regression applies linear regression techniques upon the data under the assumption that the given data points are *linearly-separable*. It tries to find the logistic function (or linear boundary) that distinctively separates the given data based on the provided class. Unlike ordinary regression, Logistic Regression outputs the probability that a given belongs to a certain class.

Inference of classification model Logistic Regression model is based on [22][23]:

$$z = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_n x_n \tag{2.1}$$

The variable *z* is called log odd which is equivalent to exponential function of linear regression, The variable β_0 is called the *intercept* which provides the value when the predictors are equal zero. $\beta_1 \cdots \beta_n$ are called the regression coefficients, $x_1 \cdots x_n$ are called the predictors. The log odd is the log likelihood and it is mapped to the interval [0,1] using (Equation 2.2) to get the probability

$$P(c_0|x_1 \cdots x_n) = \frac{1}{1+e^{-z}} \tag{2.2}$$

Logistic Regression performs an iterative process through the given data to find the maximum log likelihood. When successive iterations have small difference the model converges and we have the logistic function.

Data classification The classification is the simple process of plugging in the values of the predictors in the logistic function and compute the probability that the data point belongs to said class. Once the probability of one class is computed, the probability of the other class can be computed by *'1-P'*.

2.1.2.3 Support Vector Machine

Support Vector Machine is a supervised machine learning technique that strives to find the hyperplane that separates the data into 2 groups with the widest possible margin. The maximization of the gap allows for better classification performance. SVM spans the margin until points from the groups are reached. Assuming a dataset of n points in the form $(\vec{x}_1, y_1), (\vec{x}_2, y_2), \cdots, (\vec{x}_n, y_n)$, where $\vec{x}_i \, \epsilon \, \mathbb{R}$ is the input vector for *i*-th example. y_i is either +1 or -1 indicating the class to which *i*-th example belongs to, the example is said to be positive if $y_i = +1$ and negative when $y_i = -1$. Assuming that the points are linearly separable SVM finds the optimal hyperplane that satisfies the equation $f(x) = \vec{w}.\vec{x} + b = 0$, where $\vec{w} \, \epsilon \, \mathbb{R}$ is the weight vector and $b \, \epsilon \, \mathbb{R}$ is the bias.

In essence, multiple hyperplanes can satisfy the said equation. SVM chooses those hyperplanes as decision hyperplanes, and using these decision hyperplanes SVM tries to find the hyperplane that achieves the maximum margin as follows

$$\begin{aligned} \vec{w}.\vec{x}_i + b > 0 \qquad & for\ y_i\ = +1 \\ \vec{w}.\vec{x}_i + b < 0 \qquad & for\ y_i\ = -1 \end{aligned} \tag{2.3}$$

Assuming that *i*-th lies at the edge of the margin, therefore, the distance from the point to the hyperplane $\vec{w}.\vec{x}_i + b = 0$ is equal to $\parallel w \parallel^{-1} \mid \vec{w}.\vec{x}_i + b \mid$. SVM tries to find the optimal values for w, b that maximizes:

$$\frac{\mid \vec{w}.\vec{x}_i + b \mid}{\parallel w \parallel} \tag{2.4}$$

Since we have to maximize for two hyperplanes in positive and negative examples, the distance equation is simplified to

$$\frac{2}{\parallel w \parallel} \tag{2.5}$$

2.1.2.4 K-Means Clustering

K-Means Clustering is considered one of the most popular and commonly used clustering techniques. Given a set of points, K-Means tries to group them into clusters based on some similarity measure. The most commonly used similarity measure is the distance measure, among them the Euclidean distance is the commonly used distance formula [24].

Based on the number of classes, K-Means elects initial cluster centers, one point corresponding to each class. For the remaining available points, K-Means takes a point and calculates its distance measure with respect to all available cluster centers, and finds its nearest cluster center. K-Means assigns that point to that

cluster and updates the center center, producing a new center, by calculating the mean of the cluster member points [24].

This process is repeated for all the given points, and the final output from K-Means is the cluster centers. This way, any new incoming point can be tested against the cluster centers to find out its belongingness to what class.

2.1.3 Evidence Theory

Given a set of worlds Ω called *the frame of discernment*. One of the worlds, denoted w_0, corresponds to the actual world. An agent does not know which world in Ω corresponds to the actual world w_0. Nevertheless, the agent has some idea(opinion) about which world might be the actual one. So for every subset A of Ω, called *the focal element*, the agent can express the strength of its opinion that w_0 belongs to A. This strength is denoted *bel(A)* and called a belief function (formal definition below). Extreme values for *bel* denote full belief (1) or no belief at all (0). The larger *bel(A)*, the stronger the agent's belief that w_0 belongs to A [25][26][27][28].

2.1.3.1 Basic Belief Assignments

One of the concepts of the theory is a basic belief assignment (*bba*). Related to belief function *bel*, one can define its so-called Moebius transform, denoted *m* and called a basic belief assignment. Let

$$m : 2^{\Omega} \rightarrow [0, 1] \tag{2.6}$$

where *m(A)* is called the basic belief mass (*bbm*) given to $A \subseteq \Omega$. The value of *m(A)* represents belief that supports A - *i.e.*, the fact that the actual world w_0 belongs to A without supporting any more specific subset. In the case, when w_0

belongs to A, and nothing more is known about the value of w_0, then some part of belief will be given to A, but no subset of A will get any positive support. In that case, $m(A) > 0$ and *m(B) = 0* for all $B \subseteq A$ and $B \neq A$.

2.1.3.2 Belief Functions

The *bbm m(A)* does not quantify belief that w_0 belongs to A (*bel(A)*), . The *bbm m(B)* given to any subset B of A also supports that w_0 belongs to A. Hence, the belief *bel(A)* is obtained by summing all the *bbm m(B)* for $B \subseteq A$. At the end:

$$bel(A) = \sum_{\phi \neq B \subseteq A} m(B), \forall A \subseteq \Omega, A \neq \phi \tag{2.7}$$

$$bel(\phi) = 0 \tag{2.8}$$

The belief function *bel* satisfies the following inequality:

$$\forall n > 1, \forall A_1, A_2, \ldots, A_n \subseteq \Omega \tag{2.9}$$

$$bel(A_1 \cup A_2 \cup \ldots A_n) \geq \sum_i bel(A_i) - \sum_{i>j} bel(A_i \cap A_j) \cdots - (-1)^n bel(A_1 \cap A_2 \cap \ldots A_n) \tag{2.10}$$

As such, the meaning of these inequalities is not obvious except when n = 2. These inequalities generalize the idea that agent's belief that the actual world belongs to $A \subseteq \Omega$ can be larger than the sum of the beliefs the agent gives to the elements of a partition of A.

2.1.3.3 Combination of Two Belief Functions

Suppose there are two distinct pieces of evidence Ev1 and Ev2 produced by two sources of information. Let *bel1* and *bel2* be the belief functions induced by each

piece of evidence. These two belief functions, with the focal elements A_i and B_j respectively, may be combined into a new belief function using Dempsters rule of combination. The rule specifies the combined belief mass, *m*, assigned to each focal element C_k, where *C* is the set of all subsets produced by *A* and *B*, The rule is:

$$m(C_k) = \frac{\sum_{A_i \cap B_j = C_k; C_k \neq \phi} m(A_i)m(B_j)}{1 - \sum_{A_i \cap B_j = \phi} m(A_i)m(B_j)} \tag{2.11}$$

where the focal elements of $bel_1 = A = A_1, ..., A_i$ and $bel_2 = B = B_1, ..., B_j$. The combination of two belief functions is also know as taking the orthogonal sum, $\bigoplus$, and is written as

$$bel_3 = bel_1 \oplus bel_2 = (m(C_1), \ldots, m(C_k)) \tag{2.12}$$

The meaning of distinct for two pieces of evidence has been left undefined. It lacks rigorous definition. Intuitively it means the absence of any relation. In this case the belief function *bel2* induced by the second source is not influenced by the knowledge of the belief function *bel1* induced by the first source and vice versa

2.1.4 Transferable Belief Model

Dempster-Shafer theory has been used to develop the TBM. The model represents quantified beliefs and is based on belief functions *bel*. Given a belief function *bel*, a probability function is generated that is used to make decision by maximizing expected utilities. It requires the construction of the betting frame, *i.e.*, a list of alternatives on which the bet must be made.

Let BetFrame denotes the betting frame. The granularity of BetFrame is such that if by necessity two alternatives are not distinguishable from a

consequence-utility point of view, than they are pooled into the same granule. Once the betting frame is determined, the *bbms* are transformed by the so-called pignistic transformation into the pignistic probabilities $BetP : 2^{\Omega} \rightarrow [0,1]$ with:

$$BetP(A) = \sum_{X \subseteq BetFrame, X \neq \phi} \frac{m(X)}{1 - m(\phi)} \frac{\#(A \cap X)}{\#(X)}, A \subseteq BetFrame, A \neq \phi \quad (2.13)$$

#(X) is the number of granules of the betting frame BetFrame in X, and *m()* is called the basic belief mass. The pignistic probability function BetP is a probability function which represents the additive measure needed to compute expected utilities when decision must be made, given someone's beliefs as *bel*

2.2 Related Work

There are many academic articles and research work that relate to one or more topics of our proposed work. Our work focuses on intrusion detection, and how Machine Learning techniques can be used to detect intrusions. Within Intrusion Detection there are multiple research topics; anomaly intrusion detection, signature intrusion detection, and hybrid intrusion detection. Some researchers use the term hybrid anomaly detection when several techniques are used in anomaly intrusion detection. Others use a similar term – hybrid signature detection – when multiple techniques are used in signature intrusion detection. Basically, hybrid intrusion detection means the integration of both anomaly and signature intrusion detection within a single detection engine, whether multiple techniques are used in both or not.

In this section we are going to mention several research works that relate to this topic. First and foremost, we need to have network data and it has to be described by several features (or attributes). Whether the research involves

signature, anomaly, or hybrid intrusion detection, multiple techniques can be applied to this data. The application of techniques solves the problem of intrusion detection in the form of a classification problem. In anomaly intrusion detection we classify normal and anomaly network traffic. In signature intrusion detection we distinguish between multiple attack classes. In hybrid intrusion detection, we are doing both tasks, *i.e.*, differentiating between normal and anomaly, and once an anomaly is detected we try to determine if this anomaly is in fact an attack. Most Intrusion Detection researchers use ready-to-use datasets and apply multiple combinations and permutations of Machine Learning and CI techniques to classify network traffic

The authors in [29] proposed a new signature-based multi-layer distributed intrusion detection system using mobile agents. Their main idea is to distribute multiple agents across several layers, each holding a small database of frequent attacks. The implementation of this model detect attacks faster than having a central database of signatures. They used **Snort** IDS and used several attack tools to simulate attacks. Their proposed experiment included training and operation phases. R. Chetan et al. [30] used KDD dataset and association rule mining to implement aq Data Mining Based Intrusion detection system application prototype using .NET. Their system consists of 5 main modules; *Sensor*, *Extraction, transformation and load (ETL) Centralized data warehousing*, *Automated rule generation*, *Real-time and offline detection*, and *Report and Analysis*. The **sensor** captures stream network traffic and passes it to the **ETL** for features extraction. Of course, they did not actually use such modules in their research because they used the ready-to-use KDD dataset which contains the formatted data and its features. The **Automated rule generation** performs association rule mining technique to infer attack rules. They conclude that their proposed system offers multiple advantages of integrating individual

components, security, scalability, and high availability.

[31] proposes a hybrid intrusion detection methodology using Hidden Markov Model (HMM) and Sequence TimeDelay Embedding(STIDE). They used KDD 98 dataset and excluded 3 attacks out of 55 attacks. They compared their proposal against K-NN and proved that their proposed hybrid-HMM is better using ROC curve – False Positive Rate (FPR) and Detection Rate (DR). [32] has also found that applying a hybrid approach of combining Hidden Markov Model (HMM) with C5.0 to enhance intrusion detection. The main idea was to check whether the combination of C5.0 with HMM will produce better detection than HMM's. The hybrid combination provides a 90% improvement for all classes (attacks+normal) compared to the individual use of HMM. They used KDD data and partitioned it into 20 smaller datasets whose sizes range from 5-100% of the original KDD dataset. Each small dataset is divided further into 75% and 25 % training and test data which were used to train C5.0 and HMM and produce their respective models and then test the generated model.

The study in [33] proposes a multilayered database intrusion detection system in big data transactions. It constructs profiles of normal user queries issued against the database during the training phase, the profiles holds user-query rules that span transaction, inter-transaction, log files, and database applications source code. Then during the detection phase, the constructed profiles and their associated rules are used to differentiate between normal and abnormal user queries. Mingjun Wei et al. [34] proposed an improvement to K-means clustering algorithm via spanning tree to detect anomalies. They used KDD dataset and focused their anomaly research on the 98.3% normal data. They applied both K-Means and their proposed improved K-Means in anomaly detection. With a lower number of clusters (44 clusters), improved K-Means provided better detection rate.

The research in [35] explained in a literature manner their idea to implement Granular Intrusion Detection systems (GIDS), one GIDS to learn normal and abnormal behavior of each protocol via data mining and machine learning techniques. They state that they have implemented anomaly behavioral modules for TCP, UDP, IP, and DNS.

The research in [36] suggests a two-stage database intrusion detection system using Dempster-Shafer's theory to detect intrusive and suspicious transactions. They used lower and upper thresholds to investigate the deviations of incoming transactions as a first stage. If a transaction is found to be intrusive or suspicious, misuse detection is applied along with inter-transactional and intra-transactional features in the second stage. The authors analyzed their systems's performance by comparing it to other previously published systems. G. Folino et al. [37] argue a distributed intrusion detection framework that uses specialized ensembles and non-trainable, genetic programming-based combination function to combine ensembles. They use KDD dataset and compared their work to Greedy-Boost algorithm. They proved the validity of their work by using precision and recall metrics. The results obtained by M. Nour et al. in [38] suggest that LR produces better classification results than Naïve Bayes (NB) and Expectation Maximization clustering (EM), the results were confirmed by calculating precision and recall against UNSW-NB15 and the NSLKDD datasets. They applied a hybrid methodology that combines 1)Central Points of attribute values method (CP) and 2)Association Rule Mining (ARM) to select the highest ranked attributes before applying the techniques.

J. Kevric et al. [39] has also found that the combination of random tree and NBTree algorithms using the sum rule scheme produces better classification accuracy (89.24%) than individual algorithms, by using NSL-KDD dataset in their work. N. Haq et al. [40] studied an anomaly detection framework against

NSL-KDD dataset and showed that hybrid feature selection and ensemble base classifiers; Bayesian Network, Naïve Bayes, and J48 Decision Tree produce lower false positive rate of 0.021 than individual classifiers. They used Best First, Genetic Search, and Rank Search to select 12 important attributes. Eventually, models from base classifiers were combined using majority vote.

The reduction of false positives in intrusion detection systems was experimentally proved by [41] via cascading several data mining techniques. The author used SVM, DT, and Naïve Bayes against KDD dataset. She added an extra binary attribute to KDD dataset to label the data records as either normal or anomaly. KDD dataset is split to 66% training data and 34% test data. After training the classifiers and building models, the data is subjected to SVM then DT, and finally NB. The model performed with an overall accuracy of 99.62% and false positive rate of 1.57%. The authors in [42] proposed an event stream database based architecture to detect both signatures and anomalies. The traffic is saved to a database an the processing of intrusion detection is done in the form SQL queries. The system consists of four modules; *Data splitter*, *Event Labeler*, *Detection Module*, and *Rules Modules*. The **Data Splitter** splits streams according to; *Physical attributes*, *Communication attributes*, *Packet content*, and *Network and Flow characteristics*. The **Event Labeler** tags events based on; *Thresholds*, *Stochastic*, and *Pattern* triggers. The *Detection Module* detects **Signatures** by using **SQL queries** and **Anomalies** via **Clustering** techniques to find similar events. Based on the alerts provided by the other modules, the **Rules Module** a high level filtering layer for experts to add If-then rules to tag streams.

Chapter 3

Network Traffic Analysis

In this chapter we explain and clarify some networking aspects related to Intrusion Detection, which eventually led to our cornerstone contribution, *i.e.*, multi-perspective description of network traffic. Within the framework of the thesis, we detect intrusions; both attacks and anomalies. Therefore, we need to investigate their effects and traces within a network traffic. We start with researching attacks, Section 3.2, their definitions and execution patterns of which have already been identified by security experts. Our work on interpretation and understanding of attacks have allowed us to fully analyze canonical attack signatures, and propose multiple updates of these signatures.

Also, we want to know how an attack pattern influences and represents itself in terms of a network traffic. Therefore, we analyze several intrusion detection datasets, Section 3.3. This allows us to better understand attack execution patterns and known how to describe these patterns in terms of network traffic features. All this has led to design a multi-perspective feature-based description of network traffic, Section 3.4. The description is the basis for our proposed network anomaly profile that has the capability to represent different normal network behaviors, Section 3.5.

3.1 Classification of Network Traffic

In terms of Intrusion Detection, network traffic patterns can be classified into three classes; normal, anomaly, and attack. Figure 3.1 represents the distinction between the classes. **Circle A** represents the domain of normal traffic patterns, while $\bar{A}$, ***i.e.*, Anomaly = 1 - A**, represents anomalous traffic patterns. Within $\bar{A}$, **Circle B** represents the class of attack traffic patterns.

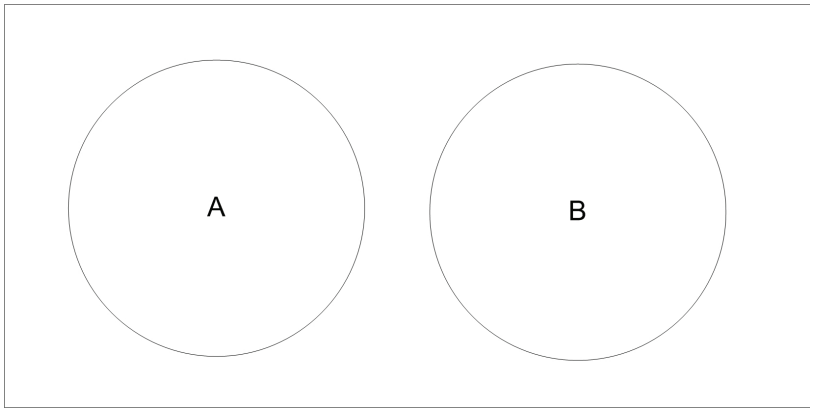


Figure 3.1: Network Traffic Classes.

However, in the case of real network traffic, the distinction between the three classes is not that crisp. If our focus is to identify normal traffic only then we need to identify thresholds of normal network behaviors (***i.e.*, Circle A**). We construct anomaly profile using the identified thresholds (the boundary of **Circle A**) and use Anomaly ID to distinguish between normal and anomalous (not-normal) network behaviors. In other words, Anomaly ID identifies any network traffic that lies outside the boundaries of **Circle A** as an anomaly. Meanwhile, **Circle B** represents attack behaviors. Signature ID identifies any network traffic that lies outside the boundaries of **Circle B** as a normal traffic.

As the result, any network traffic that 'belongs' to the area outside the circles A and B is suspicious. It is like that, because Anomaly ID treats this as an

anomalous behavior while Signature ID regards it as a normal one, at the same time. Consequently, this particular network traffic represents either a case of a new normal behavior or a new attack behavior. Such a situation is an example of a sophisticated attack we call hereafter an 'elusive' attack. These types of attack are executed in a such way that their traffic does not conform to any attack traffic, and at the same time seems to be a normal, or slightly different from normal traffic. A security administrator or an implemented IDS mistakenly considers these intrusions as normal behaviors.

3.2 Analysis of Attacks

The research focuses on seven attacks that form our set of selected attacks (Section 2.1.1.5.2). We have analyzed them from different perspectives and identified that any attack involves **one** of the following:

1. An individual protocol;
2. Multiple protocols of the same TCP/IP layer;
3. Multiple protocols that belong to several TCP/IP layers.

This implies that features of a attack signature have to represent information about specifics of network traffic observed from several perspectives/points of view. In this section, we discuss the canonical attack signatures and their respective shortcomings. Identification of these shortcomings results in updated signatures. An update of a given attack signature proposed by us may include:

1. Different/More specific values of features of the canonical attack signatures;
2. Additional features added to the canonical attack signature.

The justification of the proposed updates mainly relates to:

1. Issues related to a current topology of network and/or
2. Experience of security administrators.

Each analyzed attack is presented in the following way: firstly, we describe the canonical signature of a given attack; secondly, we identify its limitations and present a new, updated version of the signature.

3.2.1 ICMP Flood Attack

ICMP Flood is a DOS attack, Section 2.1.1.5.2, in which a target machine is overwhelmed with ICMP ping request packets, Figure 3.2.

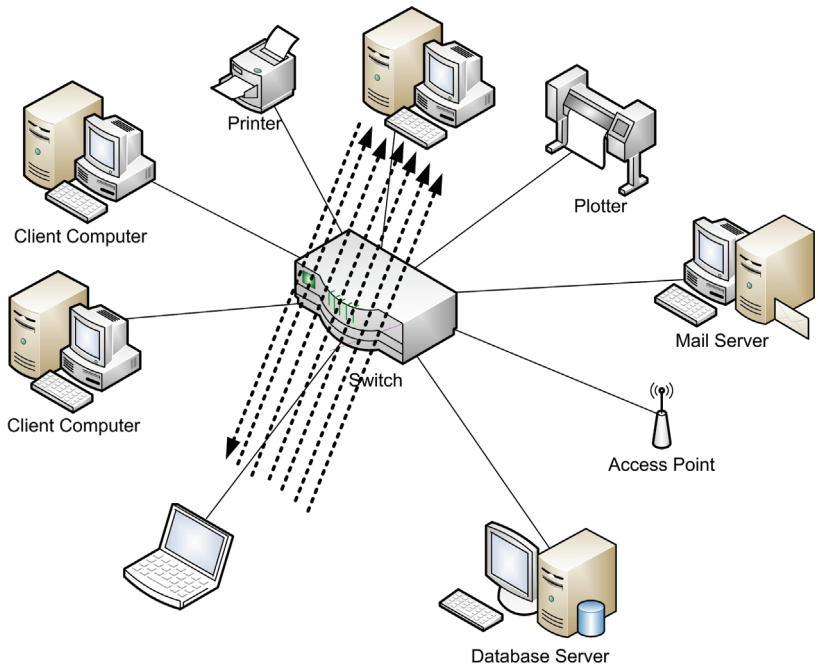


Figure 3.2: ICMP Flood Attack.

3.2.1.1 Canonical ICMP Flood Signature

The attack targets CPU cycles of a target machine by sending bursts of ICMP ping request packets over a small amount of time [10]. After a while, the target machine 'starves' and starts dropping packets, in other words, it can not accept any more packets. According to [10], the only identifiable attack signature feature is *Number of transmitted ICMP request packets between two distinct hosts.* Also, it states that the attack can be identified by observing 1000 packets per second transmitted per two distinct hosts. Table 3.1 illustrates ICMP Flood canonical signature.

Signature Feature	Feature Value
Attacker	1 Machine
Targets	1 Machine
No. of ICMP request packets per two distinct hosts	1000 packets/sec

Table 3.1: Canonical ICMP Flood Signature.

3.2.1.2 Updated ICMP Flood Signature

An important question that should be asked here is related to a number of sent packets: what if we found 750 ICMP ping request packets transmitted between two hosts in one second? would that constitute an ICMP Flood attack? A human administrator would be inclined to regard this traffic as an attack. However, if the canonical signature is implemented as a part of signature IDS, such an example of a traffic would be regarded as normal. Therefore, we propose two updates:

a. **More Specific Feature Values**: A security administrator can adjust the signature to detect 250 ICMP packets as ICMP Flood attack instead of 1000 packets. This enables the administrator to have some form of early warning indicator of attack.

b. **Additional Features**: The canonical definition of the attack does not

mention anything related to connections established between two hosts. The question here is: does all packets in a burst belong to the same connection, or do they belong to several connections? In order to determine suitable values regarding a number of established connections, we consider two issues:

- According to the specifications of Microsoft Windows operating system, we propose to follow the traffic pattern of the Windows ping utility to assign valid feature values; the Windows' ping utility issues a ping request, from one host to another, that consists of 8 ICMP ping request packets that belong to 4 ICMP connections; thus, we can assume that 8 ICMP ping packets issued to a certain host and 4 established ICMP connections with that host are 'normal' values for ICMP-related packets and connections; important note: these packets and connections are issued/established sequentially not concurrently.
- Also, specifications of Microsoft Windows operating system state that older versions of Windows allow 10 concurrent connections as the connection queue size, while newer versions allow 20 concurrent connections.

Therefore, if the observed burst of ICMP ping request packets belong to several connections, then the attack is consuming the connection queue of the target machine.

Finally, we propose the new, updated ICMP Flood attack signature, Table 3.2.

Signature Features	Feature Values
Attackers	1 machine
Target Machines	1 machine
No. of transmitted ICMP request packets per two distinct hosts	$8 \leqslant packets \leqslant 250$
No. of established ICMP connections per two distinct hosts	$4 \leqslant connections \leqslant 20$

Table 3.2: Updated ICMP Flood Signature.

3.2.2 IGMP Flood Attack

Likewise, IGMP Flood attack is a DOS attack that floods a target machine with IGMP unicast packets [10]. The attack aims to consume the CPU cycles of the target machine. After reviewing multiple publications, we did not find any explicit definition of signature for this attack. Therefore, we propose this attack signature that is similar to the one of ICMP Flood. We propose 250 IGMP unicast packets as the value of the feature: No. of transmitted IGMP unicast packets per two distinct hosts, Table 3.3.

Signature Features	Feature Values
Attackers	1 machine
Target Machines	1 machine
No. of transmitted IGMP unicast packets per two distinct hosts	$\geqslant 250 packets$

Table 3.3: Update IGMP Flood Signature.

3.2.3 Smurf Attack

Smurf attack, Section 2.1.1.5.2, is a DOS attack in which a target machine is flooded with burst of ICMP ping reply packets transmitted from multiple hosts [10][13][14][18].

3.2.3.1 Canonical Smurf Signature

The canonical signature of the attack implies that an attacker sends a malformed ICMP ping request packet to the network broadcast address; the malformed packet is designed such that the source IP Address is the address of the target machine. Unlike ICMP Flood, the target machine is flooded by several sources not just one source, and the used packets are ICMP ping reply, not ping request packets. To Summarize, the canonical attack signature can be identified as in Table 3.4.

Signature Features	Feature Value
Attacker	*m* machines
Targets	1 Machine
Overall Number of transmitted ICMP reply packets	*undefined*

Table 3.4: Canonical Smurf Signature

Where *m* represents the number of machines in the network.

3.2.3.2 Updated Smurf Signature

The canonical Smurf attack signature does not mention anything about the overall number of ICMP reply packets directed towards a target machine. Additionally, it even does not provide an average number of reply packets exchanged between distinct hosts, from which we can deduce the overall number of flood packets. Besides, like ICMP Flood, the Smurf canonical signature fails to mention anything about ICMP connections and consumption of a connection queue of the target machine (as explained in Section 3.2.1: ICMP Flood attack). Thus, we propose the following signature updates:

a. **More Specific Feature Values**: It is normal to observe ICMP ping request packets with or without corresponding reply packets; thus, it is logical and protocol-wise legitimate to assign a value of 250 packets to ICMP Flood updated signature. But, it is highly illogical and unorthodox to assign such a value for an attack whose signature depends entirely on ICMP ping reply packets. *i.e.*, we will not legitimately observe ICMP ping reply packets without their respective requests. To solve this situation, we abide by the value,8 packets, given by the traffic of Windows ping utility.

b. **Additional Features**: Based on the provided above explanations regarding the nature of Smurf attack, we propose to add packet-oriented and connection-oriented features.

To summarize, we propose the following updated Smurf attack signature, Table 3.5:

Signature Features	Feature Values
Attackers	m machines
Target Machines	1 machine
No. of transmitted ICMP reply packets per distinct hosts	$\geqslant 8$
No. of established ICMP connections per distinct hosts	$\geqslant 4$
Overall Number of attack packets	$1 * m \leqslant packets \leqslant 8 * m$
Overall Number of attack connections	$1 * m \leqslant connections \leqslant 4 * m$

Table 3.5: Updated Smurf Signature.

Where m is # of machines in network

3.2.4 LAND Attack

LAND attack, Section 2.1.1.5.2, is another DOS attack in which a target machine floods itself. An attacker sends a malformed TCP SYN packet to the target machine whose source and destination IP addresses are the target machine, and with the same source and destination ports. The target machine responds to the packet and does not realize that it recursively calls itself, thus flooding itself [10][13][14][18].

3.2.4.1 Canonical LAND Signature

From the attack definition, the canonical LAND attack signature is:

Signature Features	Feature Values
Attacker	1 Machine
Targets	1 Machine
Protocol	TCP
Overall Number of observed attack packets	*undefined*

Table 3.6: Canonical LAND Signature

3.2.4.2 Updated LAND Signature

Regardless of the used Transport protocol, it is IPv4-wise illegitimate to observe packets with the same source and destination addresses within the network.

Therefore, it is logical to assume that any number of packets – even one packet – that match this pattern, is LAND attack.
Like ICMP Flood and Smurf, the canonical LAND signature failed to mention any details regarding established connections, since TCP SYN packets are used to execute LAND attack and in the same time used to initiate connections. Thus, the attack would exhaust the capacity of a connection queue of the target machine.

a. **More Specific Feature Values**: Any number of packets more than 1.

b. **Additional Features**: 20 or more connections.

The updated LAND attack signature is presented in Table 3.7:

Signature Features	Feature Values
Attacker	1 Machine
Targets	1 Machine
Protocol	TCP
Overall Number of attack packets	$\geqslant 1$
Overall Number of attack connections	$\geqslant 20$

Table 3.7: Update LAND attack signature.

3.2.5 IPSweep Attack

IPSweep attack, also known as Ping sweep or ICMP ECHO, is a Probe attack that sweeps IP addresses within a certain range with ICMP ping request packets [10][13][14][18][21], Figure 3.3. The main goal of this attack is to identify on-line machines in a network via observing replies to ICMP ping requests.
This attack is used to gain information about a number of devices existing in the network. Though the basic definition of the signature does not provide a value for the number of destination addresses to be swept, it does say that the attack targets the whole subnet to which an attacker is connected. To put this into a context, we refer to Networking and Subnetting. There are three classes of the IP addresses: Class A that supports up to 16 million addresses per network;

Class B provides 65 thousand addresses per network; and Class C has 254 unique addresses per network.

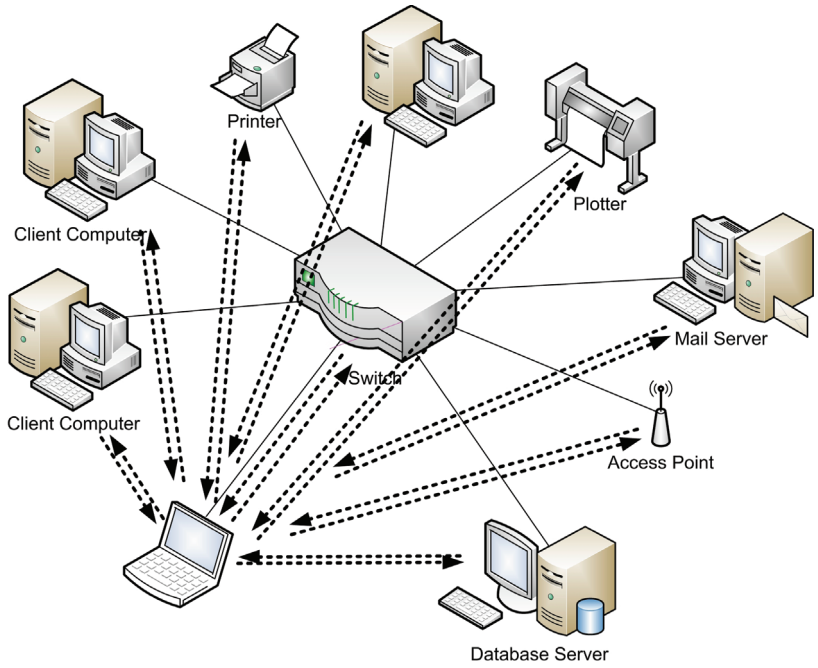


Figure 3.3: IPSweep Attack.

3.2.5.1 Canonical IPSweep Signature

This exploration activity results in the following attack signature:

Signature Features	Feature Values
Attackers	1 Machine
Target Machines (sweep range)	m machines, *i.e.*, ICMP destinations
Overall Number of transmitted ICMP packets by the attacker	*undefined*
Number of transmitted ICMP packets per distinct hosts	*undefined*

Table 3.8: Canonical IPSweep Signature.

Where m is dependent on the subnet class

3.2.5.2 Updated IPSweep Signature

The references we have investigated do not provide any quantitative information about the attack signature. We propose two updates due to some deficiencies in the definition of the attack signature.

a. **More Specific Feature Values**: The values we propose are related to two new features added to the signature, please see below. We follow the traffic pattern of Windows ping utility to assign valid values of the new features. In addition, from a normal usage of ARP protocol we know that a machine communicates with another machine using 1 to 3 packets.

b. **Additional Features**: We propose adding new features that provide additional information about ICMP connections and ARP protocol.

   - **ICMP Connections**: In the same manner as ICMP Flood, Smurf, and LAND, the IPSweep signature fails to provide any information about ICMP connections. Therefore, we propose features that represent information about established connections by an attacker, and their number per a single host.

   - **ARP Protocol**: The canonical definition of the attack does not even mention anything about ARP packets. In essence, an entire class of packets is removed from the attack signature [43]. This allows us to propose additional new signature features.

     There is a relationship between ICMP and ARP protocols. For a machine to communicate with another machine, it needs to know two addresses: physical and logical. ARP protocol is used to map a logical address to its corresponding physical address. The attacker's machine, before issuing an ICMP request, sends an ARP request to match the

logical address to its physical one.

In a nutshell, it seems that detecting IPSweep attack could be even more efficient monitoring ARP protocol than just monitoring ICMP one.

As the result, we propose updated IPSweep attack signature, Table 3.9

Signature Features	Feature Values
number of attackers	1 machine
number of target machines (sweep range)	m machines, *i.e.,* ARP and ICMP destinations
overall number of transmitted ICMP packets by an attacker	$1 * n \leqslant packets \leqslant 8 * n$
number of transmitted ICMP packets per distinct host	$1 \leqslant packets \leqslant 8$
overall number of established ICMP connections by an attacker	$1 * n \leqslant packets \leqslant 4 * n$
number of established ICMP connections per distinct hosts	$1 \leqslant packets \leqslant 4$
overall number of transmitted ARP packets by an attacker	$1 * m \leqslant packets \leqslant 3 * m$
number of transmitted ARP packets per distinct host	$1 \leqslant packets \leqslant 3$

Table 3.9: Updated IPSweep Signature.

3.2.6 InsideSniffer Attack

InsideSniffer attack is a Probe attack used to provide an attacker with the machine's name based on its IP address. Basically, the attacker captures the packets within the network, extracts their source and destination IP addresses, and matches these addresses to their respective names.

To do that, the attacker has to use a name resolution protocol, *e.g.,* DNS. The legitimate usage of name resolution protocols is to resolve a name associated with a given IP address. Thus, the attack execution pattern uses name resolution protocols illegitimately.

3.2.6.1 Canonical InsideSniffer Signature

The canonical signature states that the attacker uses DNS protocol to carry out an attack. Nothing has been mentioned about a number of resolved IP addresses. Thus the canonical signature is to search for any DNS query that tries to match an IP address to its name, Table 3.10.

Signature Features	Feature Values
number of attackers	1 machine
DNS Resolution	IP Address to Name

Table 3.10: Canonical InsideSniffer Signature.

3.2.6.2 Updated InsideSniffer Signature

While DNS is considered the primary name resolution protocol, there are other secondary name resolution protocols that offer the same functionality, *i.e.*, NBNS and LLMNR. Thus, we propose to update attack signature by monitoring name resolutions activities carried out by NBNS and LLMNR besides DNS. The proposed InsideSniffer attack signature is presented in Table 3.11.

Signature Features	Feature Values
number of attackers	1 machine
DNS Resolution	IP Address to Name
NBNS	IP Address to Name
LLMNR	IP Address to Name

Table 3.11: Updated InsideSniffer Signature.

3.2.7 PortScan Attack

PortScan attack, also known as Nmap, is a Probe attack used to scan ports of the target machine, and to see which ports are opened and closed. The literature review related to this attack [13][14][18] states that simulations of this attack were performed on 3 to 100 ports per machine. It seems the signature of this attack does not need any update. The original signature is as follows:

Signature Features	Feature Values
number of attackers	1 machine
number of target machines	1 machine
number of scanned ports	3 - 100

Table 3.12: Updated PortScan Signature.

3.2.8 Summary and Recommendations

The performed analysis of attacks has provided us with a better understanding of their specifics. We have gained knowledge what aspects of network traffic have to monitored and what values of these aspects are indictors of execution of a given attack. These aspects include: source addresses, destination addresses, packets, connections, ports, and protocols' details, frequency of their occurrences, and temporal analysis of this information. In addition, we have been able to identify layers and protocols that are targeted by the attacks:

Attack	Target Layer	Target Protocol
ICMP Flood	Internet	ICMP
IGMP Flood	Internet	IGMP
Smurf	Internet	ICMP
LAND	Transport	TCP
IPSweep	Network Interface Internet	ARP ICMP
InsideSniffer	Application	DNS NBNS LLMNR
PortScan	Transport	TCP

Table 3.13: Attack Target Layers and Protocols.

3.2.8.1 Attack Severity

Most of the literature discussing Intrusion Detection handles it from an atomic perspective. The atomicity lies in the explanation of one attack instance belonging to one cyber attack, executed by one attack source (aside from Smurf). In real life scenarios, we can find many attacks happening concurrently in any given time. A single attack source can execute multiple attacks either sequentially or concurrently. Multiple attack sources can participate in an individual attack scenario. Also, these attack sources can execute several cyber attacks (more than 1 cyber attack). Moreover, these attacks can - at the same time - target multiple machines, or they can target the same machine(s) consecutive number of times. To elaborate, each *attack scenario* can be described as a general equation.

$$Attack\ Source(s)_{1-m} + Attack\ Type(s)_{1-n} \Longrightarrow Attack\ Target(s)_{1-o} \tag{3.1}$$

Where **m** is the number of attackers, **n** is the number of executed cyber attacks, and **o** is the number of targets.

The *attack equation*, (Equation 3.1), involves **three** variables; **the Attack Source**, **the Cyber Attack itself**, and **the Attack Target(s)**, the equation can indicate an atomic (one) attack instance if **m**, **n**, and **o** are all equal to one.

The **Attack Source** does not necessarily has to be one attacker, in real life scenarios there can be multiple attackers executing attacks. In some cases, one attacker can control multiple machines and use them to launch a coordinated attack. Thus, multiplicity of attack sources guarantees the success of attacks.

The **Cyber Attack** itself does not have to be one attack instance belonging to a specific cyber attack. Again, the concurrent execution of multiple attack instances belonging to several cyber attacks guarantees success.

There are many things that an attacker can **target**, the target can be a single machine or a set of machines in a network or even the entire network.

Therefore, all attack targets can be attacked by one or more attackers concurrently. Cyber attacks can also be launched in the form of stages, to coordinate planned attacks for several durations, at specific times.

To wrap up, we consider the existence of one attacker, executing one cyber attack, against one target as one attack instance with basic severity. Severity increases when more than one attack instance is found, *i.e.*, when one of the variables in the attack equation is not one (not atomic). In these situations, when the attack severity is not basic, then we consider the executed attacks as '*elusive or evasive*' attacks.

In order to handle these complex attack situations, as we will see later in Chapter 8, we strive to simplify the detection process of an elusive (or evasive) attack by detecting its basic severity level and expand from there. First we start by dissecting network traffic based on attack type. Once we extract the network

traffic of a specific cyber attack, we go deeper and investigate the attack sources, attack targets, packets, connections and duration. This technique of tracking attacks is logical, since intelligent attackers aim to avoid detection by launching evasive (or elusive) cyber attacks as follows:

1. using multiple attackers.
2. target several attack targets.
   Theses 2 points will result in an increasing number of packets and connections
3. Executing sequential and/or concurrent cyber attacks.
4. Elongating or shortening the expected duration of attacks.

3.3 Analysis of Available Datasets

In this section we describe the available datasets used for development of instruction detection system found in the literature. We analyze them in order to better understand intrusion detection data. In general, we can identify a number of difficulties in constructing good quality intrusion detection datasets:

1. ensuring balanced data, i.e., starve to achieve a comparable percentage of normal and anomalous traffic data;
2. avoiding time-related errors that occur during traffic generation processes and result in missing data;
3. labeling data records accurately;
4. preserving confidentiality of network within which the experiments are conducted.

We provide description of three datasets: KDD, NSL-KDD, and TUIDS.

3.3.1 KDD CUP '99

The data of KDD CUP '99 was captured within the network of an Air Force Base. The data records are formatted in a connection-wise format that represents the connections initiated between two distinct network hosts. Each data record is described by 41 attributes (*i.e.*, features). These attributes are either continuous or categorical in nature. Also, the attributes are categorized into 4 different categories: basic, content, traffic, and host [10][13][14][18]:

Basic Features this category includes attributes that can be extracted directly from TCP/IP connections, *i.e.*, protocol header fields.

Traffic Features this category mainly includes time-wise attributes based on a 2 seconds window; its features are viewed from two perspectives: same host and same service. *Same Host Features* calculate connections established with the same host in the last 2 seconds; likewise, *Same Service Features* imply the number of connections established using the same service within the last 2 seconds. Primarily, these features are implemented to ease the detection of Probe and DOS attacks.

Logically, it is a correct assumption; in DOS attacks, an attacker floods the target machine with many connections that use the same service in a small amount of time. Similarly in Probe attacks, an attacker scans the target for information, this reconnaissance activity implies the use of many connections belonging to different services.

Content Features this category includes attributes suggested by the domain knowledge; knowledge like number of failed logins, administrator issued

connections, and attempt to elevate user privileges to root, *etc.* Thus, it seems that attributes of this category are implemented to ease the detection of R2L and U2R attacks.

Host Features this category includes attributes that describe the behavior of individual source hosts, from a time-independent perspective. More Likely, the attributes constructed in this category act as **plan B** for *Traffic* and *Content* features. In other words, if an attack is not detected by its intended constructed category (included features), this category detects anomalous behavior of individual hosts in an attempt to match this anomaly to an attack.

KDD CUP '99 dataset was constructed to contain both normal and attack network traffic. The traffic of 54 attacks was simulated with multiple tools during the capture process; the attacks belong to 4 categories according to the taxonomy presented in Section 2.1.1.5. Thus, the data records are labeled as normal or attack type.

Finally, limitations and criticisms on this dataset have been presented in [43], the criticisms include the attack taxonomy, synthesized data, evaluation, and results.

3.3.2 NSL-KDD

The Network Security Lab in University of New Brunswick produced NSL-KDD dataset; it is a refined version of KDD CUP '99. The authors in [15] demonstrated the deficiencies in KDD CUP '99 regarding; redundant records and difficulty levels of the dataset. Also, they provided highlights about the generated data, whether normal or attack data, and percentages of executed attacks.

Their work was based on the criticisms mentioned in [43], they claim that the presence of redundant data is a strong reason for biasing the capabilities of

machine learning techniques. In order to produce NSL-KDD they performed 2 operations; removing redundant records, and labeling the obtained refined dataset.

They reduced KDD train set from 4,898,431 records to 1,074,992 (78.05% reduction) and KDD test set from 311,027 records to 77,289 (75.15% reduction), thus selecting distinct records only. To label the obtained refined data correctly, they randomly selected 3 subset, each containing 50,000 records; afterwards, they used 7 machine learning techniques (J48 decision tree, Naïve Bayes, NBTree, Random Forest, Multi-layer perceptron, Support Vector Machine, and Random Tree) upon these datasets. As a result, they obtained 21 learned machines with 21 predictions for each data record.

3.3.3 TUIDS

One of the first research studies that aimed to propose new features other than using the internationally accepted features of KDD benchmark is TUIDS dataset [10][17]. The research conducted at Tezpur University in India produced 2 versions of TUIDS dataset; Packet Level and Flow Level TUIDS.

An isolated testbed network was set up, 16 attacks were launched using 1 commercial tool (Nmap) and a lot of user-developed tools (c-programmed). Also, normal traffic was generated. The resultant network traffic was captured and preprocessed to produce:

1. 50 feature-based packet level TUIDS dataset
2. 24 feature-based flow level TUIDS dataset, it was collected by extracting unidirectional sequence of packets that pass through a selected observation network node during a given time duration.

Also, we can conclude from TUIDS that its features belong to 4 categories; 1) Basic 2) content-based 3) Time-based, and 4) Connection-based.

3.3.4 Summary and Recommendations

From the discussion of datasets we understood how network traffic can be represented in the form of features. Also, we learned how normal and attack traffic can be represented using the same features. We determined that the same feature can be used to represent normal or attack traffic via different values. Aside from feature values, some features uniquely identify normal traffic, and others distinctively describe attack traffic.

Moreover, we grasped that features represent multiple traffic granularities like traffic protocol, packets, connections, *etc*. Furthermore, we gained knowledge about the data type of features; continuous and categorical. Besides, we learned that researchers classify features into categories based on their calculation process; basic, content-wise, time-wise, and connection-wise.

3.4 Multi-Perspective Description of Network Traffic

The thorough analysis of attacks (Section 3.2) as well as the available datasets (Section 3.3) has led us to re-visiting a way how network traffic is represented. This resulted in a new multi-perspective description of network traffic. Complexity of traffic, involvement of multiple protocols, hosts and temporal aspects of all these activities should be considered in order to properly grasp and understand behavioural patterns of normal activities of users, and activities of attackers who try to penetrate a network and stay undetected.

Data packets can be considered the basic unit of network traffic: *"All packets*

that flow in the network are sent from several sources and are received by multiple destinations". This statement alone is an indication that a network traffic should be 'seen' as least from *three* network traffic perspectives: the ***packet*** perspective, the ***scope*** perspective, and the ***address*** perspective.

We learned from the analysis of attacks and intrusion detection datasets that time is an important factor influencing a network analysis process. Thus, our *fourth* perspective is the ***temporal*** perspective.

If we move our point of interest upwards from packets, we look at groups of packets sent from sources to destinations that constitute connections between hosts. So, the *fifth* perspective is the ***connection*** perspective. On the other hand, shifting our attention towards the content of packets, each packet encapsulates several protocol headers, consequently, each protocol belongs to a certain TCP/IP layer. Therefore, the *sixth* perspective is the ***protocol*** perspective, and the *seventh* perspective is the ***layer*** perspective.

This perspective-related view allows us to represent normal and anomalous network traffic patterns in a more comprehensive way. The proposed description has two novel points:

- all perspectives are tightly interrelated;
- features we use for description of network traffic belong to multiple perspectives simultaneously.

In general, the perspectives can be categorized into *three* main perspectives: layer, protocol, and scope; and *four* auxiliary perspectives: packet, address, connection, and temporal. It is difficult to demonstrate the relationships between perspectives because of their multi-level and multi-directional interrelations. Thus, we use Figure 3.4 to demonstrate the relations between main and auxiliary perspectives.

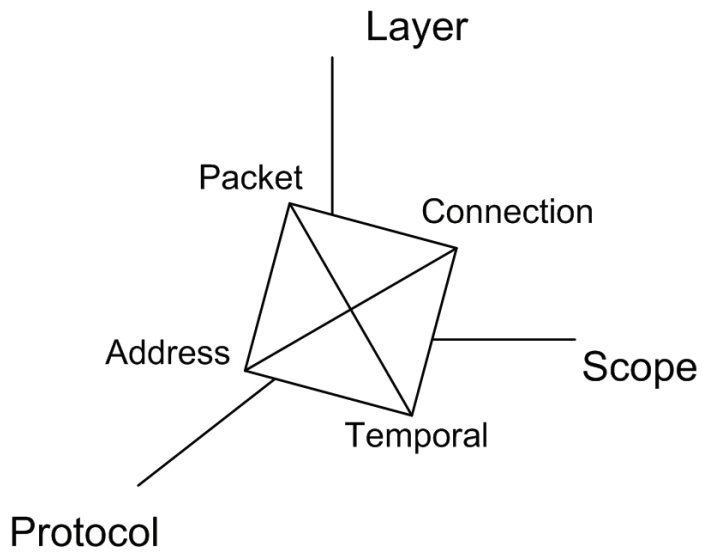


Figure 3.4: Illustration of Multi-Perspective Description of Network Traffic.

3.4.1 Packet Perspective

Packet perspective includes features that represent counts of different types of packets. In the context of the main perspectives we can distinguish the following categories of packet feature:

Protocol Perspective: features representing packet counts of a specific protocol, *e.g.*, *TCP packets*;

Scope Perspective: features representing packet counts at network-level and host-level; network-level packet counts relate to overall number of packets transmitted through the whole network, *e.g.*, *network IGMP packets*; while host-level packet counts include a number of packets sent or received by a specific host, *e.g.*, *UDP packets of address 192.168.20.221*.

In addition, more specific number-related features focus on packet counts of a specific type (*i.e.*, requests or replies); of a specific protocol, or scope, *e.g.*, *ICMP*

query request packets of address 192.168.10.1.

Finally, all of these features can be determined over a certain time unit, *i.e.*, seconds, minutes, hours, days, or a user-defined duration, and produce packet rates per time unit, *e.g.*, *ARP packets per seconds*.

3.4.2 Address Perspective

The address perspective includes features that stand for numbers of source or destination addresses. Address counts are useful to investigate a number of sources or destinations within a specific network traffic pattern. The relation to the main perspectives results in:

Protocol Perspective: features with distinct source and destination address counts for a specific protocol, *e.g.*, *TCP source addresses*;

Scope Perspective features with network-level distinct source and destination address counts, *e.g.*, *UDP destination addresses*; and features with host-level address counts that signify:

- a number of destinations contacted by a specific source address, *e.g.*, *ARP destination addresses that 192.168.1.22 contacted*;
- a number of sources communicating with a certain destination address, *e.g.*, *ICMP source addresses communicated with 192.168.1.10*.

Like in the case of packet perspective, any of these features can be calculated over certain time units.

3.4.3 Scope Perspective

Network traffic can be observed at a network level that includes traffic among all machines/hosts connected to a given network, or at a host level that includes

traffic directed to and from an individual host. We grasp this concept in our proposed network traffic representation as the scope perspective. We propose network-level features, *e.g., network IGMP packets*, and host-level features, *e.g., UDP packets of address 192.168.20.22*. The importance of this perspective comes from the fact that anomalies existing within the network on a global view are caused by traffic related to one or more hosts (local view).

3.4.4 Temporal Perspective

Analysis of network traffic takes into account a temporal aspect of occurring events. Therefore, the proposed description includes temporal versions of features. Temporal perspective features can be classified into:

- time-dependent features;
- time-independent features.

Time-dependent features are designed and implemented based on a selected time unit, *e.g.*, they represent values of features per second, minute, hour, day, or user-specified duration. Thus, it is possible to associate any feature with time to obtain rates per time unit, or to obtain valuess within a certain time interval, *e.g., ICMP packets per minute = 3.6*.

Time-independent features are those features that do not relate to a specific time unit.

3.4.5 Connection Perspective

Network hosts exchange several packets that are related to different protocols. In this context, groups of packets are associated with connections: a connection is a set of packets exchanged in a specific order to fulfill a specific task. The order and

content of the packets are fully governed by a given communicating protocol. The concept of connection is found in three specific places within the TCP/IP suite. We recognize ICMP connections, TCP connections, and applications-oriented UPD connections – the type we proposed and explained in Section 5.3. Processes of monitoring and analysis of a network traffic from the point of view of connections lead to determining a very important aspect of *connection status*.

The features that belong to the connection perspective are:

Protocol Perspective numbers of ICMP, TCP, and UDP connections;

Scope Perspective number of connections over the entire network, also number of connections associated with a local host:

- connections established by a certain source host, *e.g., TCP connections of 192.168.10.10*
- connections established to a certain destination host, *e.g., ICMP connection to 192.168.10.10*

In addition, several features representing connection counts with respect to specific connection statuses can be calculated. Similar to the packet and address perspectives, any of these features can be also calculated for specific time units.

3.4.6 Protocol Perspective

Different layers of specific protocol stacks contain variety of protocols. Therefore, we propose a number of features describing network traffic from the protocol perspective. For the TCP/IP suit of protocols, we represent a network traffic in for form of:

1. ARP features, *e.g., ARP request packets per second*

2. ICMP features, *e.g., ICMP Error packets*
3. IGMP features, *e.g., IGMP Multicast packets*
4. TCP features, *e.g., TCP connections*
5. UDP features, *e.g., UDP Packets*
6. DNS features, *e.g., DNS Unicast destination addresses*
7. NBNS features, *e.g., NBNS Multicast Packets*
8. LLMNR features, *e.g., LLMNR request packets of 192.168.10.10*

This classification contains just a few most popular protocols. By all means, it is not a full list of protocols that a system administrator/security specialist could be interested in. This list is a set of protocols used for our case study presented in this thesis.

3.4.7 Layer Perspective

Data packets hold in their payload protocol headers. Each header represents a specific protocol that is specific to a certain TCP/IP layer. Layer-related features can represent information about layer's characteristics that is protocol-dependent, or represent the protocol-independent layer's characteristics. From the layer perspective, features representing a network traffic can be as follow:

1. Network Interface layer features:
    a. Protocol-specific features, *e.g., ARP response packets per minute*
    b. Layer-specific features

2. Internet layer features:

    a. Protocol-specific features, *e.g.*, *ICMP query packets*

    b. Layer-specific features, *e.g.*, *LAND Indicator*

3. Transport layer features:

    a. Protocol-specific features, *e.g.*, *TCP Services*

    b. Layer-specific features, *e.g.*, *Transport Connections Count*

4. Application layer features:

    a. Protocol-specific features, *e.g.*, *DNS request packets*

Each group of features contains protocol-specific features and layer-specific features. Protocol-specific features are associated with protocols of TCP/IP architecture. Layer-specific features can be joint or disjoint. Joint layer-specific features describe aspects of a layer derived based on several protocols of the layer, while disjoint layer-specific features describe the layer's aspects that are protocol independent. A list of proposed features can be found Appendix C.

3.5 Network Anomaly Profile

The proposed multi-perspective description of network traffic can be treated as a comprehensive representation of traffic. It could be used to represent normal as well as anomalous network status, network conditions under attacks, or be applied for constructing detailed descriptions of attacks – attack signatures. In this subsection, we illustrate application of the proposed description to express a network anomaly profile.

The proposed network anomaly profile contains **189** features. Those features represent **four** layers of TCP/IP architecture and **eight** different protocols. As explained in the previous section, features represent *layer-based* characteristics, *protocol-based* traits, and views for a specific *scope*. Additionally, there are features from the **four** auxiliary perspectives: *connection, packet, temporal, and address*.

For example, a feature called '*ARP Packets Per Second*' is a number of transmitted ARP packets through the network per second. This feature is considered a layer-based feature that describes the network access layer. At the same time, it is a protocol-based feature that outlines a trait of ARP protocol. Scope-wise, it defines network behavior related to ARP packets. Finally, it is considered time-based feature whose value is measured with respect to specific time unit – seconds.

A profile is created by processing the data collected during a number of normal sessions (our implemented tool allows a user to select number of sessions from 2 to 10). In the example shown below, we created a network anomaly profile using ten normal sessions. Table 3.14 and Table 3.15 represent profile features related to ARP and ICMP protocols respectively. The feature values are determined based on simple processing of network traffic observed during the specified (ten in our case) sessions. The calculated values are treated as thresholds, *i.e.*, if a value of a given feature is below a threshold - it is recognized as normal traffic, otherwise it is considered an anomaly.

As we can see in Table 3.14, there are many types of threshold values, integers, floats, and ranges. Feature ID 1 is '*ARP packet count*' and it represents the range of number of ARP packets. To calculate this range, we determine the number of ARP packets for each session, and then take the maximum and minimum values. The *four* perspectives of this particular feature are: layer, protocol, scope, and

packet.

ID	Layer: **Network Interface** Protocol: **ARP**			
	Scope	Auxiliary Perspectives	Feature Name	Feature Value (threshold)
1	Network	Packet Temporal	Packet Count	[350,2406]
2			Packets Per Second	0.4
3			Packets Per Minute	24.2
4			Packets Per Hour	1451.3
5			Request Packets	[235,1328]
6			Requests Per Second	0.2
7			Requests Per Minute	12.1
8			Requests Per Hour	723.9
9			Response Packets	[115,1135]
10			Responses Per Second	0.2
11			Responses Per Minute	12.1
12			Responses Per Hour	726.8
13		Address	Source Addresses	7
14			Destination Addresses	8
15			Request Source Addresses	5
16			Request Destination Addresses	8
17			Response Source Addresses	6
18			Response Destination Addresses	4
19	Host	Address	Destinations Per Source	[1,6]
20		Packet	Requests Per Destinations	[1,365]
21		Packet	Overall Requests Per Source	[1,892]
22		Packet	Responses Per Sources	[1,353]
23		Packet	Overall Responses Per Destination	[4,582]

Table 3.14: Network Anomaly Profile: ARP Protocol Sample Features

Feature ID 3 is '*ARP packets per minute*', we calculate ARP packets per minute rate for each session. Then we average the obtained rates over the number of sessions. This feature relates to *five* perspectives; layer, protocol, scope, packet and temporal. Feature ID 13 is '*Source Addresses*', it represents a distinct number of source addresses within ARP packets of the chosen sessions. This feature also represents *four* perspectives; layer, protocol, scope, and address, it does not represent temporal perspective because it is a time-independent feature. Similarly, feature ID 19 is '*Destinations per Source*', we calculate a number of destinations contacted by a source IP address using ARP packets across all sessions. Afterwards, we inferred the minimum and maximum values. The feature is associated with *four* perspectives; layer, protocol, scope, and address. Likewise, Table 3.15 below represents network anomaly profile features for ICMP protocol, and we will notice that the table includes some features that relate to

the connection perspective.

For some of the protocols mentioned in Section 3.4.6 (most commonly used protocols), we designed and implemented their corresponding features within our network profile. All of the points represented in this chapter represent our initial research steps, later in our work we used these vital points and expanded them to complete out thesis context.

ID	Layer: **Internet** Protocol: **ICMP**			
	Scope	Auxiliary Perspectives	Feature Name	Feature Value (threshold)
1	Network	Packet Temporal	Packet Count	[0,425]
2			Packets Per Second	0
3			Packets Per Minute	3.2
4			Packets Per Hour	189.6
5			Error Packets	[0,417]
6			Error Per Second	0
7			Error Per Minute	3.1
8			Error Per Hour	188
9			Query Packets	[0,8]
10			Query Packets Per Second	0
11			Query Packets Per Minute	0
12			Query Packets Per Hour	1.6
13			Query Request Packets	[0,4]
14			Query Request Packets Per Second	0
15			Query Request Packets Per Minute	0
16			Query Request Packets Per Hour	0.8
17			Query Response Packets	[0,4]
18			Query Response Packets Per Second	0
19			Query Response Packets Per Minute	0
20			Query Response Packets Per Hour	0.8
21		Address	Source Addresses	2
22			Destination Addresses	2
23			Request Source Addresses	1
24			Request Destination Addresses	1
25			Response Source Addresses	1
26			Response Destination Addresses	1
27		Packet	Maximum Packet Size	84
28		Connection	Connection Count	[0,2]
29		Connection Temporal	Connections Per Second	0
30			Connections Per Minute	0
31			Connections Per Hour	0.2
32	Host	Address	Destinations Per Source	[0,1]
33		Packet	Requests Per Destinations	[0,4]
34			Overall Requests Per Source	[0,4]
35			Responses Per Sources	[0,4]
36			Overall Responses Per Destination	[0,4]
37		Connection	Connections Per Destination	[0,2]
38			Overall Connections Per Source	[0,2]

Table 3.15: Network Anomaly Profile: ICMP Protocol Sample Features

3.6 Updated Network-Tailored Attack Signatures

Now that we understand how to update attack signatures, Section 3.2, and now we have the maximum boundary of normal network behavior, Section 3.5, we thought to integrate both of these findings to produce *Updated Network-Tailored Attack Signatures*. Updated network-tailored attack signatures are those that fit a specific network, in which features are guided by the maximum normal thresholds. In this section, we are going to match attack signature features to their corresponding network anomaly profile features. The matching provides us with more secure signatures.

3.6.1 Updated Network-Tailored ICMP Flood Signature

A quick matching between ICMP Flood signature features (Section 3.2.1 and network anomaly profile: ICMP protocol features (Section A.1.2) indicate that feature number 3 in the signature corresponds to feature number 33 in ICMP protocol features, also feature 4 in the signature corresponds to feature 37 in the profile. Since these features have the same meaning we assigned their related values to produce the signature in Table 3.16

ID	Signature Features	Feature Values
1	Attackers	1 machine
2	Target Machines	1 machine
3	No. of transmitted ICMP request packets per distinct hosts	4
4	No. of established ICMP connections per distinct hosts	2

Table 3.16: Updated Network-Tailored ICMP Flood Signature.

3.6.2 Updated Network-Tailored IGMP Flood Signature

In the same manner, from our network anomaly profile: IGMP protocol features (Section A.1.3), we find that the normal behavior of the network does not contain any unicast communication with IGMP protocol, Table 3.17.

ID	Signature Features	Feature Values
1	Attackers	1 machine
2	Target Machines	1 machine
3	Number of transmitted IGMP unicast packets per distinct host	0

Table 3.17: Updated Network-Tailored IGMP Flood Signature.

3.6.3 Updated Network-Tailored Smurf Signature

Again, we can match Smurf attack features to our network anomaly profile: ICMP protocol features (Section A.1.2), to produce updated network-tailored Smurf attack signature, Table 3.18. Also, if we incorporate the network topology within the signature, we have 8 machines in our network, thus the variable m renders tobe 8 machines.

ID	Signature Features	Feature Values
1	Attackers	m machines
2	Target Machines	1 machine
3	No. of transmitted ICMP reply packets per distinct hosts	$1 \geqslant 4$
4	No. of established ICMP connections per distinct hosts	$1 \geqslant 2$
5	Overall Number of attack packets	$8\ (1*8) \leqslant packets \leqslant 32\ (4*8)$
6	Overall Number of attack connections	$8\ (1*8) \leqslant connections \leqslant 16\ (2*8)$

Table 3.18: Updated Network-Tailored Smurf Signature.

Where ***'8'*** is the number of machines in the network.

From the table, it is obvious that signature features are now more concrete and secure.

3.6.4 Updated Network-Tailored LAND Signature

We can match LAND attack signature feature to features 1 our network anomaly profile: Network Interface Layer features (Section A.1.4), to produce updated network-tailored LAND attack signature, Table 3.19.

Another addition to the signature is to free the signature from using ONLY TCP protocol. Since we updated the signature with connection features beside packet

features, then the attack can be executed by the attacker and detected by the implemented IDS using any type of connection.

ID	Signature Features	Feature Values
1	Target Machines	1 machine
2	Overall Number of attack packets	Any
3	Protocol	TCP, UDP, or ICMP
4	Overall Number of attack connections	$\geqslant 0$

Table 3.19: Updated Network-Tailored LAND Signature.

3.6.5 Updated Network-Tailored IPSweep Signature

We can match IPSweep attack signature feature to features from our network anomaly profile: ARP and ICMP Protocol features (sections A.1.1 and A.1.2 respectively), to produce updated network-tailored IPSweep attack signature, Table 3.20. The table illustrates the updated network-tailored IPSweep signature, taking into consideration the network topology of 8 machines.

ID	Signature Features	Feature Values
1	Attackers	1 machine
2	Target Machines (sweep range)	7 (6+1) machines
3	Overall Number of transmitted ICMP packets by the attacker	> 4
4	Number of transmitted ICMP packets per distinct hosts	> 4
5	Overall Number of transmitted ARP packets by the attacker	$1 \leqslant packets \leqslant 18\ (3 * 6)$
6	Number of transmitted ARP packets per distinct hosts	$1 \leqslant packets \leqslant 3$
7	Overall Number of established ICMP connections by the attacker	$2\ (2 * 1) \leqslant packets \leqslant 14\ (2 * 7)$
8	Number of established ICMP connections per distinct hosts	$2\ (2 * 1) \leqslant packets \leqslant 14\ (2 * 7)$

Table 3.20: Updated Network-Tailored IPSweep Signature.

The main thing to notice here is the number of destination that constitute the sweep range. According to our network anomaly profile, the maximum number of ARP destinations is 6 and the maximum number of ICMP destinations is 1, and this is the main feature that should be tracked and detected. Because an attacker can use several (ARP and/or ICMP) packets to scan machines but still, the governing feature is the number of scanned destinations, not the number of packets and connections as in DOS attacks.

3.6.6 Updated Network-Tailored InsideSniffer Attack

Concerning network topology, no further updates can be provided for this attack other than what is explained in Section 3.2.6

3.6.7 Updated Network-Tailored PortScan Attack

PortScan attacks deals with one machine communicating with another machine using multiple ports. As explained in sections 3.2.7 and 2.1.1.5.2 the attacker tries to scan open and closed ports of target machine via communication. The optimum number of port which can be considered normal can be seen in our network anomaly profile: TCP and UDP protocol features. Any communication between 2 hosts that encounters more services than 7 (5 for TCP and 2 for UDP) is considered suspicious. Thus, we proposed to use this number in network-tailored PortScan signature.

ID	Signature Features	Feature Values
1	Attackers	1 machine
2	Target Machines	1 machine
3	Scanned Ports	7

Table 3.21: Updated Network-Tailored PortScan Signature.

Chapter 4

Proposed System

In this chapter, we introduce and briefly describe the proposed *Two-stage Hybrid Intrusion Detection System*. The main goal of the system is to detect intrusions whether they are anomalies or attacks. We use both intrusion detection methodologies and implement them as two modules: Anomaly ID, and Signature ID.

In a nutshell, our system detects anomalies within a network traffic at first. Then, it performs further analysis of the traffic to check if the detected anomaly conforms to a known attack.

Here, we provide an overview of the system and its components, Section 4.1, and clarify some architectural aspects of its features, Section 4.2.

4.1 System Overview

The description of the proposed system starts with a clarification and demonstration of the system's control sequence and data flow. Figure 4.1 illustrates the overall view of our system. As it can be seen, the system consists of three modules:

1. Network Traffic Collecting and Processing Module, Section 4.1.2;

2. Anomaly Intrusion Detection Module, Section 4.1.2;

3. Signature Intrusion Detection Module, Section 4.1.3.

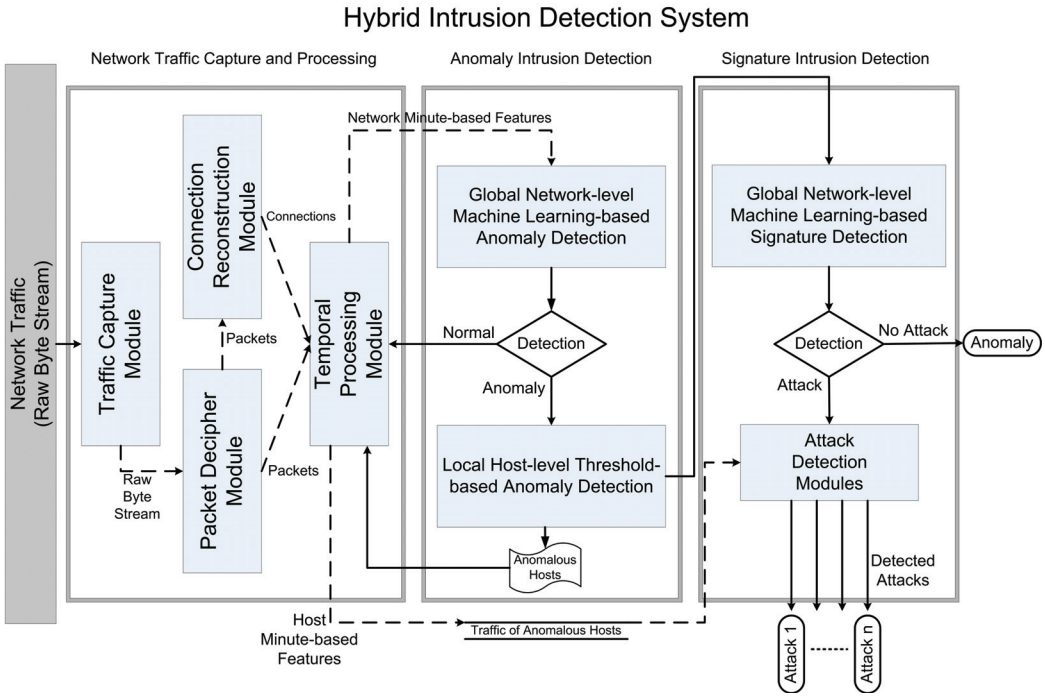


Figure 4.1: Overall System Overview.

Let us describe functionality and present some details of each of the modules.

4.1.1 Network Traffic Collecting and Processing Module (NetDataCoP)

This module captures and processes a network traffic. It uses multiple procedures and algorithms to extract and generate features from the collected data. The resulting features describe a current state of network. They are features of our proposed multi-perspective network traffic description. They are calculated on a temporal basis and illustrate a traffic over a specified period of

time, could it be a minute, two minutes or ten minutes whatever the system is configured to do. In our work, the feateures are calculated over a period of one minute. Therefore, we often refer to this features as *minute-wise features*.
A description of full functionality of this module is provided in Chapter 5. Here, we include a very brief overview of the module's functionality and demonstrate an interaction between its parts.

NetDataCoP consists of *four* sub-modules:

1. Traffic Capture sub-module (explained in Section 5.1) captures network traffic from the IDS network card in the form of stream of raw data; the data is passed to the Packet Decipher sub-module for processing;

2. Packet Decipher sub-module (explained in Section 5.2) accepts the incoming byte streams and interprets them; 'builds' packets out of them, as well as extracts related protocol headers and fields; the interpreted packets are passed to the Connection Reconstruction sub-module;

3. Connection Identification and Reconstruction sub-module (explained in Section 5.3) identifies and reconstructs ICMP and TCP connections; it also investigates UPD connections, i.e., finds and analyzes UDP logical connections based on the captured packets; the interpreted *packets* and reconstructed *connections* are forwarded to the Time-wise Processing sub-module;

4. Network Traffic Time-wise Processing sub-module (explained in Section 5.4) generates temporal-based features representing a condition of captured traffic over a specified period of time; once these features are calculated, they are passed to the next stage of our system: Anomaly Intrusion Detection Module.

4.1.2 Anomaly Intrusion Detection Module

The calculated features describing the network traffic are received by the Anomaly ID engine to ascertain whether the current state of network is normal or anomalous. As we have mentioned earlier, the features are calculated over a period of one minute, and they are called *minute-wise features*. If the traffic is classified as normal, no further processing occurs – the NetDataCoP continues collecting network data. If the traffic is classified as anomalous, the *Signature Intrusion Detection Module* is activated. Anomaly Intrusion Detection Module contains two main sub-modules:

1. Global Network-level ML-based Anomaly Detection, Section 7.1.

2. Local Host-level Threshold-based Anomaly Detection, Section 7.2.

The current state of network is investigated globally at a network level, i.e., a full network traffic data is used. The generated traffic features constitutes an input to multiple classifiers. Each classifier predicts a current state of the network, and all predictions are aggregated using elements of Evidence Theory. If the predicted state of the network is anomalous (not normal) the *Local Host-level Threshold-based Anomaly Detection sub-module* is initiated.

Local Host-level Threshold-based Anomaly Detection infers host-centric traffic features from the anomalous traffic data. The traffic associated with each host in the network is compared against the previously determined host-based feature thresholds. These thresholds represent maximum values of features representing a normal traffic/behaviour of individual hosts. This step is intended to gather information about the detected anomaly including:

1. hosts related with the detected anomaly;

2. a TCP/IP layer where the anomaly occurred;

3. a protocol associated the anomaly.

The collected information about the anomaly helps in its further analysis whether it is an actual attack or just an anomaly. If the detected anomaly is classified by the Signature ID engine as an attack, the collected information helps identifying details of the attack and the attacker. If the detected anomaly is not an attack, then the collected information helps security administrators in better understanding details of the anomaly.

4.1.3 Signature Intrusion Detection Module

The minute-wise features representing the current traffic are passed to this module when Anomaly ID classifies the current state of the network as anomalous. Signature ID investigates whether the anomalous data contains information about one or more attacks. If attacks exist then the detected attacks are reported to security administrators. If no attack exists then the current traffic is deemed as anomaly. After reporting, the Signature ID awaits for the next anomalous data to investigate. This module includes two main sub-modules:

1. Global Network-level ML-based Signature Detection, Section 8.1.

2. Signature Detection/Recognition Modules, Section 8.2.

The detection of attacks within the anomalous data takes place as a two step process. Firstly, global evidences are investigated to signify the existence of one or more attacks. This step is performed by subjecting the traffic to classifiers representing pre-defined attack models.

Once global traces of attacks are detected, the data is passed to *Signature Detection/Recognition Module* for recognizing signatures of attacks. This module processes the traffic collected over a period of time, i.e., minute-wise features in

our case. It analyzes the data from the packet and connection perspectives. It searches for the attack signature within the traffic and gathers an attack-related information.

4.2 Architectural Aspects

It is clear by now that the basic idea behind distinguishing a normal traffic from an anomalous traffic, and an anomalous traffic from an attack traffic is related to the generated network traffic features. Within our proposed system, network traffic features constitute basic units of knowledge. We can expand and generalize these units into normal or anomalous views, or attack signatures. This knowledge is organized in a hierarchical way. Let us have a closer look at this hierarchy from top to bottom. We describe knowledge of anomalies in Section 4.2.1, while of signatures in Section 4.2.2.

4.2.1 Anomaly ID

We distinguish between normal and anomalous traffic using features from a scope-centric perspective, i.e.: Global Network-level and Local Host-level. These scopes are represented as two large containers in Figure 4.2.

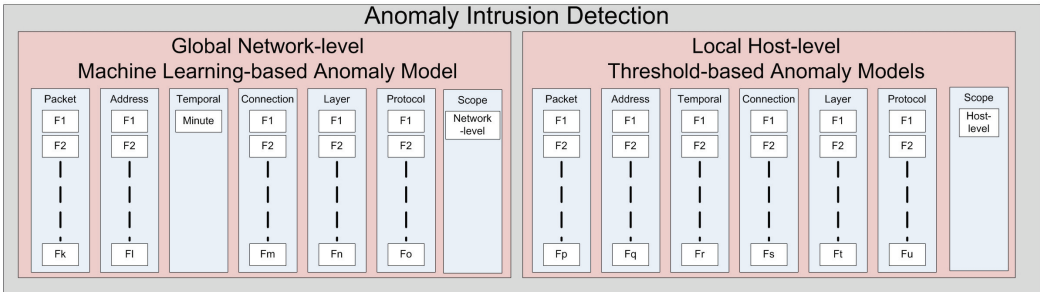


Figure 4.2: Architectural Aspects: Anomaly Detection.

Global Network-level knowledge contains features that belong to multiple perspectives from our proposed multi-perspective network traffic description. Each perspective is represented as a 'vertical' rectangular container that contains a number of individual features seen in the figure as small boxes. In the presented work, the temporal perspective of network traffic is a single minute-wise representation. For that reason, there is only one feature box in the temporal perspective container with the label *minute*. Also, the scope perspective contains only one feature, i.e., Network-level.

Local Host-level knowledge is similar to the Network-level in having features and perspectives. Also, the scope perspective contains only one feature, i.e., Host-level.

The collection of features that belong to several perspectives represent our overall knowledge about anomalies.

4.2.2 Signature ID

For signature detection, the system is able to identify several attacks. As in the case of anomaly detection, it is done using two groups of features from a scope-centric perspective, i.e.: Global Network-level and Local Host-level. However this time, we substitute the perspective contexts in each group with attack contexts that include distinctive features related to each individual attack, Figure 4.3. Each attack context is represented in the figure as a large container, while individual features are shown as small boxes within the containers. The labels of the those boxes are different for each attack context, since each attack is represented/described by its own features.

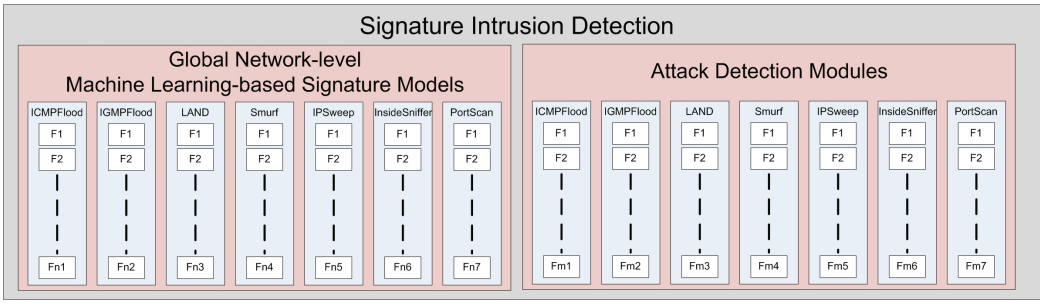


Figure 4.3: Architectural Aspects: Signature Detection.

In Global Network-level, like in Anomaly ID, attacks are detected using in a minute-wise representation of network traffic data. This means, the Global Network-level performs its task of distinguishing attacks using attack signatures represented from a single temporal and a single scope perspectives. In Attack Detection/Recognition Modules, the traffic data associated with a host that has been identified as anomalous in Anomaly Intrusion Detection module is extracted from a network traffic data. An attack signature is detected in the data from the perspective of packets and connections. In other words, the system analyzes the packets and connections of the anomalous host and tries to match its traffic patterns to attack signatures. Therefore, we can say that this investigation scheme implicitly includes three perspectives: scope, packet, and connection, alongside the attack contexts.

Chapter 5

Network Data Collecting and Processing (NetDataCoP)

A good understanding of attacks (sections 3.2 and 3.6) and network traffic description (Section 3.4) allows us to focus on design and development a system for capturing traffic data and generating features that constitute the proposed multi-perspective description of network traffic.

Our intention was to use ready-to-use, free, commercial network capture tools such as Wireshark [44] and Snort [45]. We used them and tried to capture and process network data. However, the performance and obtained datasets were not suitable for our research. Both Wireshark and Snort were able to capture network packets and decapsulate protocol headers, but the tools DID NOT:

- analyze protocols that we require for our intrusion detection methods;
- provide all protocol fields within headers of protocols they analyzed;
- identify and reconstruct connections from decapsulated packets;
- perform any temporal processing of captured data.

As the result, we have designed and implemented our own packet capture

and processing software called the Network Data Collecting and Processing NetDataCoP system. It consists of four main modules:

- Traffic Capture Module, next section.
- Packet Decipher Module, Section 5.2
- Connection Identification and Reconstruction Module, Section 5.3
- Network Traffic Temporal Processing Module, Section 5.4

5.1 Traffic Capture Module

The capture module consists of a driver and the related C# wrapper. The WinPcap driver [46] is used in the implemented IDS to expand the capabilities of operating system, *i.e.*, to access TCP/IP network layer data within the captured raw byte stream traffic. The driver allowed the implemented IDS to access byte stream traffic of *Internet*, *Transport*, and *Application* layers. We linked the driver to our software for processing raw byte stream traffic. Each captured byte stream is received by IDS as a communication socket. Having many streams captured asynchronously by IDS, we process communication sockets using multi-threading technology. Each asynchronous communication socket is passed to a separate thread for processing. The processing is done via byte-wise comparison of captured byte stream against the specifications of protocol headers (details in the next section).

We also needed an access to *Network Interface* layer. For this purpose, we installed and utilized the Pcap.Net [47] wrapper. It is a .Net C++ and C# wrapper that allowed us to access Network Interface layer byte stream, *i.e.*, Ethernet Frame header. This enabled us to interpret all fields of all protocol headers

within TCP/IP protocol suite. In our work we are focus on the protocols mentioned earlier, Section 3.4. This set of protocols include the ones that are involved in execution of the selected set of attacks (Section 3.2). We study these protocols to design and implement our proposed hybrid intrusion detection system (Chapter 4).

5.2 Packet Decipher Module

Our research targets ARP and IPv4 packets only, thus we study Ethernet II frame (also known as Ethernet Version 2). It is the most commonly used type of Ethernet frames in today's networks, and it is utilized by TCP/IP suite [3][1].

5.2.1 Ethernet II Frame (Ethernet Version 2)

An Ethernet II frame is used to transport data packets across Ethernet networks within its payload, refer to Section D.1 for more information regarding its structure. Table 5.1 illustrates sample Ethernet II data that we captured during our experiments and interpreted with the developed software NetDataCoP.

EthernetID	Timestamp	Source MAC Address	Destination MAC Address	EtherType
2118640	22:27:23.380	84:7B:EB:17:CC:05	10:5F:06:F1:22:48	IpV4
2118641	22:27:23.443	84:7B:EB:17:CC:05	10:5F:06:F1:22:48	IpV4
2118642	22:27:23.487	B8:70:F4:AD:C9:3F	10:5F:06:F1:22:48	IpV4
2118643	22:27:23.613	B8:70:F4:AD:C9:3F	10:5F:06:F1:22:48	IpV4
2118644	22:27:23.773	10:5F:06:F1:22:48	84:7B:EB:17:CC:05	IpV4
2118645	22:27:23.830	10:5F:06:F1:22:48	84:7B:EB:17:CC:05	IpV4

Table 5.1: Ethernet II Header Sample Data (NetDataCoP).

5.2.2 Address Resolution Protocol (ARP)

Whenever a machine needs to send an Ethernet frame to a certain destination, it needs to know its physical address. If it is known, then the sender encapsulates

an IP datagram within the Ethernet payload along with the physical addresses of the source and destination. Conversely, if the physical address of the destination is not known, the sender machine uses ARP to map the logical address of the destination to its respective physical address [1][48][49][50][51]. For more details about ARP, refer to Section D.2. Table 5.2 illustrates sample ARP data that we captured and interpreted.

PacketID	Operation	Sender Hardware Address	Sender Protocol Address	Target Hardware Address	Target Protocol Address
2118382	Request	B8:70:F4:AD:C9:3F	192.168.1.68	10:5F:6:F1:22:48	192.168.1.254
2118383	Request	B8:70:F4:AD:C9:3F	192.168.1.68	10:5F:6:F1:22:48	192.168.1.254
2118384	Reply	10:5F:6:F1:22:48	192.168.1.254	B8:70:F4:AD:C9:3F	192.168.1.68
2118385	Reply	10:5F:6:F1:22:48	192.168.1.254	B8:70:F4:AD:C9:3F	192.168.1.68

Table 5.2: ARP Packet Sample Data (NetDataCoP).

It is evidential from Table 5.2, concerning the MAC and protocol addresses, that these packets form an ARP request-reply pattern.
Using these ARP fields as basis for feature design, now we can design and implement ARP-related features. For example, feature 13 *"Source Addresses"* from the proposed multi-perspective network traffic description (Table C.1) and the proposed network anomaly profile (Table A.1), is the number of distinct ARP source addresses. So, we calculate a value for this feature by tracking the *'Sender Protocol Address'* field within a group of ARP packets.

5.2.3 Internet Protocol version 4 (IPv4)

The Internet Protocol version 4 (IPv4) (Figure D.4) [1][52][53][54] is an unreliable connectionless datagram protocol utilized by TCP/IP to deliver packets to their intended destinations. IPv4 does not support any flow control or error control. Thus, datagrams exchanged between two distinct hosts can reach their intended destinations out of order, or they might contain some errors, or they might

even be lost along the way. For more details about IPv4, refer to Section D.3. Table 5.3 illustrates sample IPv4 data that we captured and interpreted during our experiments.

PacketID	Ver	HLEN	Service	Total length	Identification	Flags	Fragmentation Offset	TTL	Protocol	Source IP Address	Destination IP address
1501669	4	20	0	137	9883	N	0	119	17	65.55.158.118	192.168.1.68
1501670	4	20	0	137	9883	N	0	119	17	65.55.158.118	192.168.1.68
1501671	4	20	0	129	16640	N	0	4	17	192.168.1.65	239.255.255.250
1501672	4	20	0	129	16640	N	0	4	17	192.168.1.65	239.255.255.250
1501673	4	20	0	259	24352	N	0	128	17	192.168.1.71	192.168.1.65

Table 5.3: IPv4 Packet Sample Data (NetDataCoP).

Again like ARP, IPv4 fields can be used as basis for feature design and implementation. Referring back to Section 3.2.4, LAND attack is characterized by packets whose source and destination addresses are the same. So, we designed feature 53 from the proposed multi-perspective network traffic description (Table C.1) and its corresponding implemented feature 1 within the proposed network anomaly profile (Table A.4); *'LAND Indicator'*. It is a binary feature which is set to 1 in case the source and destination addresses of given packet are the same, and set to 0 otherwise.

5.2.4 Internet Control Message Protocol (ICMP)

The Internet Control Message Protocol (ICMP) compensates for the deficiencies of IPv4 by providing error control and management mechanisms [53][54][55]. There are two categories of ICMP packets: error-reporting packets and query packets. Attacks utilize ICMP query packets, 3 attacks in our selected set of attacks use ICMP Echo packets, *i.e.*, *Ping Packets*. Section D.4 includes more details about ICMP. Table 5.4 illustrates sample ICMP data that we captured and interpreted.

ID	Implemented		IPv4			ICMP			
	PacketID	Timestamp	Source IP	Destination IP	Protocol	Type	Code	Identifier	Sequence Number
1	1259667	15:13:00.133	192.168.10.5	192.168.10.3	1	8	0	3	125
2	1259668	15:13:00.143	192.168.10.5	192.168.10.3	1	8	0	3	125
3	1259669	15:13:00.150	192.168.10.3	192.168.10.6	1	0	0	5	62
4	1259670	15:13:00.160	192.168.10.3	192.168.10.6	1	0	0	5	62
5	1259671	15:13:00.170	192.168.10.5	192.168.10.3	1	8	0	46	103
6	1259672	15:13:00.177	192.168.10.5	192.168.10.3	1	8	0	46	103
7	1259673	15:13:00.187	192.168.10.3	192.168.10.5	1	0	0	3	125
8	1259674	15:13:00.197	192.168.10.3	192.168.10.5	1	0	0	3	125

Table 5.4: ICMP Sample Data (NetDataCoP).

To increase understanding to the captured ICMP data, we illustrate it with some associated fields from IPv4 header. The protocol field of value 1 indicates that the payload of IPv4 packet contains an ICMP header.

ICMP query Echo request packets have **Type** field of value **8**, while echo response packets have **Type** field of value **0**. From the figure, it is clear that multiple ICMP query Echo (*i.e.*, ping) request and response packets are sent to and from several hosts within the network.

ICMP Flood, Smurf, and IPSweep are executed using ICMP **Echo request and reply** packets. Nonetheless, other ICMP packet types can indicate errors resulting from the execution of other attacks.

Once again like ARP and IPv4, ICMP fields can be used as basis for feature design and implementation. So, the designed feature 13 from the proposed multi-perspective network traffic description (Table C.2) and its corresponding implementation within the proposed network anomaly profile (Table A.2); *'Query Request Packets'*, indicates the number of observed query request packets. Its value is calculated by tracking all ICMP packets whose **Type** field is equal to **8**.

5.2.5 Internet Group Management Protocol (IGMP)

IP communication can be either unicast communication or multicast communication. Unicast communication involves one sender and one

receiver, *i.e.*, one-to-one communication, while multicast communication is a one-to-many communication involving one sender and multiple receivers. Internet Group Management Protocol (IGMP) [1][53][54][56][57] is a protocol that manages multicast groups memberships. More details about IGMP can be found in Section D.5. Table 5.5 illustrates sample IGMP data that we captured and interpreted during our experiments. In order to provide a more illustrative meaning for the captured data. Table 5.5 shows sample IGMP data for a group of IGMP packets, which were captured and interpreted within our experiments.
We implemented the first two columns; PacketID and Timestamp, the PacketID is a unique number attached to each new incoming packet, the Timestamp is the time within which the packet was transmitted upon the network. The next three columns are interpreted from IPv4 header of the packets, and the rest of the fields are their respective IGMP header fields.

ID	Implemented		IPv4			IGMP		
	PacketID	Timestamp	Source IP	Destination IP	Protocol	Type	MRT	Group Address
1	289878	13:13:50.577	192.168.1.254	224.0.0.1	2	17	100	0.0.0.0
2	289882	13:13:52.320	192.168.1.68	224.0.0.251	2	22	0	224.0.0.251
3	289885	13:13:54.260	192.168.1.67	224.0.0.252	2	22	0	224.0.0.252
4	289889	13:13:56.990	192.168.1.67	239.255.255.250	2	22	0	239.255.255.250
5	289891	13:13:57.527	192.168.1.68	224.0.0.253	2	22	0	224.0.0.253
6	290231	13:14:54.760	192.168.1.254	224.0.0.1	2	17	100	0.0.0.0
7	290232	13:14:54.790	192.168.1.67	239.255.255.250	2	22	0	239.255.255.250
8	290629	13:15:35.633	192.168.1.68	224.0.0.252	2	22	0	224.0.0.252
9	290732	13:15:51.417	192.168.1.68	224.0.0.253	2	22	0	224.0.0.253

Table 5.5: IGMP Sample Data (NetDataCoP).

Looking at the data provides a better understanding of what the packet is intended for. Table 5.5 illustrates multiple types of IGMP packets, it includes General Query (17) and Group Membership Report (22). The protocol field from IPv4 header indicates the protocol within IPv4 payload, protocol value of 2 instructs us to look for IGMP header.
In our experiments, the address '*192.168.1.254*' represents our router and gateway, thus in rows 1 and 6 it sends General Query to group 244.0.0.1 asking

if it has contributing members. Likewise, rows 2, 5 and 8 indicate that the host '*192.168.1.68*' joined three multicast groups; 224.0.0.251, 224.0.0.253, and 224.0.0.252, respectively. We can notice that the sample data contains redundancy *i.e.*, rows 4 and 7 which indicate that the host '*192.168.1.67*' wishes to join multicast group *239.255.255.250*. This pattern is not trivial. It is our intention to illustrate such a pattern, because according to IGMP specifications the Group Membership Report has to be sent from the host two successive times. Under normal conditions, the destination addresses of IGMP packets are multicast destinations. However, in the case of IGMP Flood attack IGMP is used illegitimately to flood the target machine. Thus, the detailed examination of IGMP, one more time, contributes within the design and implementation of proposed features in our multi-perspective description of network traffic (Section 3.4), network anomaly profile (Section 3.5).

For example, feature 39 – *IGMP Packet Count* – in the proposed multi-perspective network traffic description (Table C.2) and its corresponding implementation within the proposed network anomaly profile, feature 1 (Table A.3); means the number of observed IGMP packets. Its values can be calculated by tracking IPv4 packets whose Protocol field is equal to 2. Out of these packets, further processing can be made to distinguish between multicast and unicast IGMP packets by investigating the *Destination IP* field of IPv4 header.

5.2.6 Transmission Control Protocol (TCP)

While Internet layer protocols are responsible for host-to-host communication and packet delivery, transport layer handles process-to-process packet delivery. Transmission Control Protocol (TCP) is a connection-based, reliable transport

protocol [1][58][59]. TCP connections involve the following phases that ensure their reliability:

1. establishing a connection requires interaction of both hosts;
2. exchanging data over an established connection;
3. terminating a connection with a confirmation.

More details describing TCP protocol can be found in Section D.7. A sample of TCP data is shown in Table 5.6, this data has bee captured and interpreted using NetDataCoP.

ID	Implemented		IPv4			TCP	
	PacketID	Timestamp	Source IP	Destination IP	Protocol	Source Port	Destination Port
1	289977	13:14:27.550	192.168.1.68	191.236.106.123	6	1684	80
2	289978	13:14:27.617	192.168.1.68	191.236.106.123	6	1685	80
3	289979	13:14:27.663	191.236.106.123	192.168.1.68	6	80	1684
4	289980	13:14:27.723	192.168.1.68	191.236.106.123	6	1684	80
5	289981	13:14:27.780	191.236.106.123	192.168.1.68	6	80	1685
6	289982	13:14:27.833	192.168.1.68	191.236.106.123	6	1685	80
7	289983	13:14:27.887	192.168.1.68	191.236.106.123	6	1684	80

Table 5.6: TCP Sample Data (NetDataCoP).

The displayed data illustrates an interesting scenario: a host in our network '*192.168.1.68*' is communicating with an external host using TCP, a port 80 is associated with *HTTP* application protocol. Thus, our host is invoking an HTTP protocol at the host '*191.236.106.123*'. We notice *two* different port numbers: 1684 and 1685 on our host. This means that two applications, or maybe one application with two different connections are communicating with '*191.236.106.123*' using HTTP.

The detailed interpretation of TCP header fields, again contributes within the design and implementation of proposed features. Feature 9 *"Invoked Services"* in our multi-perspective description of network traffic (Table C.3), and its corresponding implementation in the proposed network anomaly profile (Table A.5), relates to the number of invoked applications on top of TCP. Referring

to Section D.6, we differentiate between *System*, *User*, and *Ephemeral* ports within the source and destination port fields of TCP header. TCP Invoked services is the number of distinct System and User ports – as source port or destination port – within a group of given TCP packets.

5.2.7 User Datagram Protocol (UDP)

Unlike TCP, User Datagram Protocol (UDP) is a connectionless, unreliable protocol [1][60]. Applications use UDP when simplicity and speed are favoured over reliability. It is suited for sending small messages with less (lesser than TCP) interaction between the source and destination, more details in Section D.8. Table 5.7 illustrates sample UDP data that was captured and interpreted using our software NetDataCoP.

ID	Implemented		IPv4			TCP	
	PacketID	Timestamp	Source IP	Destination IP	Protocol	Source Port	Destination Port
1	289959	13:14:25.057	192.168.1.68	224.0.0.251	17	59631	5355
2	289973	13:14:27.360	192.168.1.66	239.255.255.250	17	1049	8082
3	290040	13:14:31.620	192.168.1.68	192.168.1.254	17	52753	53
4	290041	13:14:31.693	192.168.1.68	192.168.1.254	17	60553	53
5	290042	13:14:31.743	192.168.1.68	192.168.1.255	17	137	137
6	290043	13:14:31.803	192.168.1.68	224.0.0.252	17	54045	5355
7	290044	13:14:31.860	192.168.1.68	224.0.0.252	17	56874	5355
8	290045	13:14:31.923	192.168.1.68	192.168.1.254	17	60028	53
9	290046	13:14:31.977	192.168.1.68	192.168.1.255	17	137	137
10	290047	13:14:32.027	192.168.1.68	192.168.1.255	17	137	137

Table 5.7: UDP Sample Data (NetDataCoP).

We can observe that the host '*192.168.1.68*' uses unicast and multicast communication with UDP as transport protocol and multiple application protocols. UDP port 53 (rows 3, 4, and 8) is DNS protocol, port 5355 (rows 1, 6, and 7) is LLMNR protocol, and port 137 (rows 5, 9, and 10) is NBNS protocol. All these application protocols are name resolution protocols, which means that the host '*192.168.1.68*' is trying to resolve something using these name resolution protocols.

The details of UDP header fields allow us to design and implement the proposed

features for our multi-perspective description of network traffic, and network anomaly profile. An example here, like TCP, would be feature 22 *"Invoked Services"* in our multi-perspective description of network traffic (Table C.3), and its corresponding implementation in the proposed network anomaly profile (Table A.6, feature 9), is the number of invoked applications on top of UDP. Like TCP, referring to Section D.6, we differentiate between *System*, *User*, and *Ephemeral*, given a group of UDP packets, we calculate the *"Invoked Services"* by counting the distinct exitance of *System* and *User* port numbers within source and destination ports of UDP.

5.2.8 Bootstrap Protocol (BOOTP)

The Bootstrap Protocol (BOOTP) is mainly used to provide a diskless machine with its IP address and boot file. Upon boot up, since the machines do not have a hard disk to boot from, Therefore they need load their operating system from somewhere into memory. When the machine boots up it requests to know its boot file and its respective IP address. BOOTP server holds multiple boot files for several operating systems, so it sends a BOOTP reply to the diskless machine informing it of its BOOTP file and IP address.

Later and before the design of Dynamic Host Configuration Protocol (DHCP), machines that have hard disks and installed operating systems would use BOOTP to know their IP address. After DHCP, BOOTP is still used to provide functionalities for DHCP.

BOOTP utilizes two ports 68 and 67. Port 68 is used as '*BOOTP client*' while port 67 represents '*BOOTP server*'. When a client needs to use BOOTP it issues a request using '*BOOTP client*' port and sets the destination port as '*BOOTP server*' [1][61][62][63][64][65]. Table 5.8 illustrates such request-reply pattern

with sample BOOTP data. More details are in Section D.9.

ID	Implemented		IPv4			UDP		BOOTP					
	PacketID	Timestamp	Source IP	Destination IP	Protocol	Source Port	Destination Port	Identifier	OPCode	HwTYPE	HwLEN	Hops	Seconds
1	2110084	22:12:24.267	192.168.1.69	255.255.255.255	17	68	67	4211123293	1	1	6	0	0
2	2110085	22:12:24.337	192.168.1.69	255.255.255.255	17	68	67	4211123293	1	1	6	0	0
3	2110086	22:12:24.417	192.168.1.69	255.255.255.255	17	68	67	4211123293	1	1	6	0	0
4	2110087	22:12:24.497	192.168.1.254	192.168.1.69	17	67	68	4211123293	2	1	6	0	0
5	2110088	22:12:24.683	192.168.1.254	192.168.1.69	17	67	68	4211123293	2	1	6	0	0

Table 5.8: BOOTP Sample Data (NetDataCoP).

The data in the table indicates that *192.168.1.69* issued three broadcast bootp requests and our network's gateway '*192.168.1.254*' responded with two unicast bootp replies. The protocol field in IPv4 header denotes that the transport protocol is UDP. The ports 68 and 67 mean that the application protocol is BOOTP. The distinction between BOOTP requests and replies is obvious due to the *OPCode* field, *HwType* of value 1 means *10mb Ethernet*, and *HwLEN* shows a 6-byte length of hardware address, *i.e.*, *MAC Address*.

5.2.9 Domain Name System (DNS)

The Domain Name System (DNS) protocol is used to map host names to their respective addresses [1][5][66][67][68][69]. Tables 5.9, 5.10, and 5.11 illustrate sample DNS packets that we captured during one of our experiments and analyzed with NetDataCoP. DNS uses port number 53, more detailed description of the protocol is provided in Section D.10.

Serial	Implemented		IPv4			Transport	
	PacketID	Timestamp	Source IP	Destination IP	Protocol	Source Port	Destination Port
1	290040	13:14:31.620	192.168.1.68	192.168.1.254	17	52753	53
2	290041	13:14:31.693	192.168.1.68	192.168.1.254	17	60553	53
3	290045	13:14:31.923	192.168.1.68	192.168.1.254	17	60028	53
4	290054	13:14:32.473	192.168.1.254	192.168.1.68	17	53	52753
5	290055	13:14:32.567	192.168.1.254	192.168.1.68	17	53	60553

Table 5.9: DNS Sample: Internet and Transport Data (NetDataCoP).

Let us analyze the content of the tables. Table 5.9 contains the *Internet* and *Transport* data of five DNS packets. The client that issues DNS requests has the

address *'192.168.1.68'* and the DNS server's address is *'192.168.1.254'*.

The firs 3 packets are obviously DNS requests since the destination port of UDP is 53, conversely packets 4 and 5 are DNS responses as UDP source port is equal to 53. Matching the Ephemeral ports of the packets, we can observe that the last 2 packets (rows 4 and 5) are responses for the first 2 packets (rows 1 and 2), and within the displayed sample packets we do not see a corresponding response packet for packet 3. Now remember this observation and look at DNS data of these packets below to confirm our conclusion, Table 5.10.

Serial	DNS											
	ID	QR	OpCode	AA	TC	RD	RA	RCode	QDCOUNT	ANCOUNT	NSCOUNT	ARCOUNT
1	62360	0	0	0	0	1	0	0	1	0	0	0
2	10129	0	0	0	0	1	0	0	1	0	0	0
3	11206	0	0	0	0	1	0	0	1	0	0	0
4	62360	1	0	0	0	1	1	0	1	5	0	0
5	10129	1	0	0	0	1	1	0	1	5	0	0

Table 5.10: DNS Sample: DNS Protocol Data (NetDataCoP).

Table 5.10 illustrates the DNS data for the sample packets in Table 5.9. Previously we mentioned that the first 3 packets are requests and the last 2 are responses. This information can be confirmed here by looking at the values of column *'QR'*; value of 0 indicates a request packet and value 1 indicates a response packet.

Also we were able to deduce that packets 4 and 5 are responses to packets 1 and 2 respectively. This deduction is confirmed in Table 5.10 by matching packets that have the same **'ID'** value, we see that packets 1 and 4 share the same *'ID'* of 62360, therefore packet 4 is a response for packet 1. Likewise packets 2 and 5 have the same *'ID'* of 10129, thus packet 5 is a response for request packet 1.

Now let us explain the rest of the DNS data. All packets have only 1 RR entry in the question section as depicted by column **'QDCOUNT'**, the response packets (4 and 5) have additional RR entries, each have 5 RRs in their *Answer* section as indicated by **'ANCOUNT'** column. Table 5.11 illustrates the corresponding RR

details of these packets.

Serial	RRs								
	Question			Answer					
	QNAME	QTYPE	QCLASS	Name	Type	Class	TTL	RDLENGTH	RDATA
1	aspnet.uservoice.com	1	1	–	–	–	–	–	–
2	aspnet.uservoice.com	28	1	–	–	–	–	–	–
3	.azure.microsoft.com	1	1	–	–	–	–	–	–
4	aspnet.uservoice.com	1	1	aspnet.uservoice.com	1	1	738263040	4	104.16.96.65
				aspnet.uservoice.com	1	1	738263040	4	104.16.95.65
				aspnet.uservoice.com	1	1	738263040	4	104.16.92.65
				aspnet.uservoice.com	1	1	738263040	4	104.16.93.65
				aspnet.uservoice.com	1	1	738263040	4	104.16.94.65
5	aspnet.uservoice.com	28	1	aspnet.uservoice.com	28	1	738263040	16	2400:CB00:2048:1:0:0:6810:6041
				aspnet.uservoice.com	28	1	738263040	16	2400:CB00:2048:1:0:0:6810:5E41
				aspnet.uservoice.com	28	1	738263040	16	2400:CB00:2048:1:0:0:6810:5D41
				aspnet.uservoice.com	28	1	738263040	16	2400:CB00:2048:1:0:0:6810:5C41
				aspnet.uservoice.com	28	1	738263040	16	2400:CB00:2048:1:0:0:6810:5F41

Table 5.11: DNS Sample: DNS RR Data (NetDataCoP).

Table 5.11 illustrates RR details of the DNS sample packets. We know that packet 4 is a response for packet 1, and packet 5 is a response to packet 2. DNS request packet 1 is querying host name **'aspnet.uservoice.com'** and it needs to know its respective **IPv4 Address** (indicated by the **QTYPE** field of value 1). The response to that query in packet 4 provided 5 IPv4 addresses for that host.
Likewise, DNS request packet 2 is querying the same host name but now it needs to know its **MAC address** (indicated by the **QTYPE** field of value 28). Of course, you can expect 5 different MAC addresses corresponding to the 5 IPv4 addresses.

5.2.10 Link-Local Multicast Name Resolution (LLMNR)

While DNS is considered the primary name resolution protocol, LLMNR is regarded as secondary name resolution protocol. It is quite similar to DNS and it is used whenever there is no DNS server present in a network [6]. LLMNR uses port 5355 over UDP, more details in Section D.11.
Table 5.12 shows Internet and Transport layer sample LLMNR data, which was captured and interpreted during our experiments. While Table 5.13 shows its respective LLMNR detailed data.

Serial	Implemented		IPv4			Transport	
	PacketID	Timestamp	Source IP	Destination IP	Protocol	Source Port	Destination Port
1	232935	19:04:19.3330000	192.168.1.70	224.0.0.252	17	59816	5355
2	232936	19:04:19.5030000	192.168.1.70	224.0.0.252	17	59816	5355
3	232978	19:04:27.0130000	192.168.1.70	224.0.0.252	17	51528	5355
4	232979	19:04:27.1830000	192.168.1.70	224.0.0.252	17	51528	5355
5	247255	20:01:00.4430000	192.168.1.68	224.0.0.252	17	56976	5355
6	247256	20:01:00.6270000	192.168.1.68	224.0.0.252	17	50256	5355

Table 5.12: LLMNR Sample: Internet and Transport Data (NetDataCoP).

Serial	LLMNR										
	ID	QR	OpCode	C	TC	T	RCode	QDCOUNT	ANCOUNT	NSCOUNT	ARCOUNT
1	49492	0	0	0	0	0	0	1	0	0	0
2	49492	0	0	0	0	0	0	1	0	0	0
3	28519	0	0	0	0	0	0	1	0	0	0
4	28519	0	0	0	0	0	0	1	0	0	0
5	60148	0	0	0	0	0	0	1	0	0	0
6	41651	0	0	0	0	0	0	1	0	0	0

Table 5.13: LLMNR Sample: LLMNR Protocol Data (NetDataCoP).

It is clear from the *QR* column that the packets are LLMNR requests (value 0). Also, we can notice that all packets contain 1 entry within Question section illustrated by *QDCOUNT* column. Table 5.14 illustrates the corresponding RR data of these packets.

Serial	RRs (Question)		
	QNAME	QTYPE	QCLASS
1	sysp-PC	255	1
2	sysp-PC	255	1
3	isatap	1	1
4	isatap	1	1
5	wpad	1	1
6	wpad	28	1

Table 5.14: LLMNR Sample: LLMNR RR Data (NetDataCoP).

In order to understand what was happening, we need to match the data in all three tables: 5.12, 5.13, and 5.14. From Table 5.12, the first 2 packets are requests issued by *'192.168.1.70'* to multicast address *'224.0.0.252'*. Table 5.13 indicates that they are related (*'ID'* field), also it indicates that they have only 1 question RR (*'QDCOUNT'* field). Table 5.14 shows that the 2 request packets are resolving any available information (*QTYPE* value of 255, Table D.2) about host *'sysp-PC'*

(*'QNAME'*).

The same investigative logic can be said for the rest of the displayed packets, except that the query names are different. From tbLLMNRSample3, Packets 3 and 4 are resolving *'isatap'* (Intra-Site Automatic Tunnel Addressing Protocol), while packets 5 and 6 are resolving *'wpad'* (Web Proxy Address).

As explained before, the interpretation of LLMNR header fields contribute to the design and implementation of features for the proposed multi-perspective network traffic description and network anomaly profile (features 17 – 24 Table C.4 and all the features in Table A.10.)

5.2.11 NetBIOS Name Service (NBNS)

NetBIOS Name Service protocol (NBNS), like LLMNR, is considered a secondary name resolution protocol. It is part of the NetBIOS service that can be invoked using both TCP or UDP. Like LLMNR, it can be used when DNS server is not present. Also, NBNS packets have the same structure as DNS and LLMNR concerning sections and RRs [70][7]. More details in Section D.12, Table 5.15 displays sample NBNS captured and interpreted using NetDataCoP.

Implemented		IPv4			UDP	
PacketID	Timestamp	Source IP	Destination IP	Protocol	Source Port	Destination Port
1739940	23:11:55.020	192.168.1.71	192.168.1.255	17	137	137
1739941	23:11:55.070	192.168.1.71	192.168.1.255	17	137	137
1739942	23:11:55.157	192.168.1.71	192.168.1.255	17	137	137
1739943	23:11:55.240	192.168.1.72	192.168.1.71	17	137	137
1739944	23:11:55.363	192.168.1.72	192.168.1.71	17	137	137

(a) NBNS Sample Packets: Internet and Transport Data

PacketID	NBNS												
	Name_TRN_ID	R	Opcode	AA	TC	RD	RA	B	RCode	QCount	ANCount	NSCount	ARCount
1739940	51236	0	0	0	0	1	0	1	0	1	0	0	0
1739941	51236	0	0	0	0	1	0	1	0	1	0	0	0
1739942	51236	0	0	0	0	1	0	1	0	1	0	0	0
1739943	51236	1	0	1	0	1	0	0	0	0	1	0	0
1739944	51236	1	0	1	0	1	0	0	0	0	1	0	0

(b) NBNS Sample Packets: NBNS Data

PacketID	Question RRs			Answer RRs					
	Name	Type	Class	Name	Type	Class	TTL	RDLength	RData
1739940	ECE220-PC	32	1	–	–	–	–	–	–
1739941	ECE220-PC	32	1	–	–	–	–	–	–
1739942	ECE220-PC	32	1	–	–	–	–	–	–
1739943	–	–	–	ECE220-PC	32	1	3767731200	6	192.168.1.72
1739944	–	–	–	ECE220-PC	32	1	3767731200	6	192.168.1.72

(c) NBNS Sample Packets: RR Data

Table 5.15: NBNS Sample Packets (NetDataCoP).

Table 5.15a displays the Internet and Transport data of 5 packets, Table 5.15b shows their corresponding NBNS data, and Table 5.15c illustrates the associated RRs. It is clear from the tables that the first 3 packets are NBNS requests – the value of **0** in the **'R'** field in Table 5.15b. We can also notice that they are sent to the broadcast address of the network. These 3 packets also contain 1 question RR as indicated by **'QCOUNT'** column, Table 5.15c shows the values of such RRs. By examining these 3 packets we can realize that *'192.168.1.71'* is resolving *'ECE220-PC'* to its respective name.

The last 2 packets are the response to the first 3 packets. Host *'192.168.1.72'* realized that the broadcast query involves his name, so he responded to the requests using unicast mode. We notice that these packets contain 1 Answer

RR and their relevant information is displayed in Table 5.15c. We can easily determine that these packets are responses to the first 3 packets by looking at the **'Name_TRN_ID'**. Again, the understanding of NBNS header fields helps in the design of NBNS-related features in the proposed multi-perspective network traffic description (features 9 – 16, Table C.4) and the features in the proposed network anomaly profile (Table A.9)

5.2.12 Simple Network Management Protocol (SNMP)

Simple Network Management Protocol (SNMP) is a protocol used to manage network resources. It remotely inspects or alters resource information [1][8][71][72]. SNMP utilizes port 161 and is used for communication between management station and corresponding managed agents. It works on top of 2 other protocols; Structure of Management Information (SMI) and Management Information Base (MIB). SMI defines management rules and MIB defines the entities to be managed for individual hosts. SNMP packets allow management information to be exchanged by reading objects statuses and modifying their values. More details on SNMP, SMI, and MIB can be found in Section D.13 regarding packet format, encoding rules, and encoding entities. Table 5.16 illustrates sample SNMP packets captured and interpreted within our experiments.

Implemented		IPv4			UDP	
PacketID	Timestamp	Source IP	Destination IP	Protocol	Source Port	Destination Port
164436	13:13:46.093	192.168.10.150	192.168.10.1	17	63868	161
164592	13:14:12.803	192.168.10.150	192.168.10.1	17	63868	161
165363	13:54:44.337	192.168.10.150	192.168.10.1	17	54897	161
165470	13:54:53.163	192.168.10.150	192.168.10.1	17	54897	161

(a) SNMP Sample Packets: Internet and Transport Data

PacketID	Version	Community String	PDU	Request ID	Error	Error Index	Varbind List	Varbind	Object Identifier
164436	0	public	160	1	0	0	48	48	1.3.6.1.2.1.1.5.0
164592	0	public	160	1	0	0	48	48	1.3.6.1.2.1.1.5.0
165363	0	public	160	1	0	0	48	48	1.3.6.1.2.1.1.5.0
165470	0	public	160	1	0	0	48	48	1.3.6.1.2.1.1.5.0

(b) SNMP Sample Packets: SNMP Data

Table 5.16: SNMP Sample Packets (NetDataCoP).

From Table 5.16a the usage of SNMP as destination port indicates that the packets are request packets. A *Community String* of value public means that the SNMP request packet aims for a *'read-only'* access to the object. From Table D.6 a PDU (Protocol Data Unit) of value 160 indicates that the packets are 'GetRequest PDU' packets. Varbind List (Table D.6) of value 48 indicates that the packets contain a sequence of Varbinds. A value 48 of Varbind signifies a sequence of two fields; an Object ID and its corresponding value.
An object identifier (OID) of *'1.3.6.1.2.1.1.5.0'* indicates that the request concerns the *'sysName'* of the target machine as follows

Numeric	1	3	6	1	2	1	1	5	0
Interpretation	iso	org	dod	internet	mgmt	mib-2	system	sysName	
Nominal	iso.org.dod.internet.mgmt.mib-2.system.sysName								

Table 5.17: Object Identifier Interpretation

5.2.13 Simple Service Discovery Protocol (SSDP)

The Simple Service Discovery Protocol (SSDP) also known as Universal Plug and Play Protocol (UPNP) is a network discovery protocol [9][73][74]. SSDP is multicast that uses the address '*239.255.255.250*' and port 1900. Additional

details can be found in Section D.14.

Table 5.18 illustrates sample SSDP packets, which were captured and interpreted in our experiments.

Implemented		IPv4			UDP	
PacketID	Timestamp	Source IP	Destination IP	Protocol	Source Port	Destination Port
2118631	22:27:22.560	192.168.1.64	239.255.255.250	17	53582	1900
2118632	22:27:22.600	192.168.1.64	239.255.255.250	17	53582	1900
2118633	22:27:22.640	192.168.1.254	192.168.1.64	17	1900	53582
2118635	22:27:22.717	192.168.1.254	192.168.1.64	17	1900	53582

(a) SSDP Sample Packets: Internet and Transport Data

PacketID	Packet Information
2118631	M-SEARCH * HTTP/1.1 Host: 239.255.255.250:1900 MAN:"ssdp:discover" MX:3 ST: upnp:rootdevice
2118632	M-SEARCH * HTTP/1.1 Host: 239.255.255.250:1900 MAN:"ssdp:discover" MX:3 ST: upnp:rootdevice
2118633	HTTP/1.1 200 OK CACHE-CONTROL: 1800 EXT: – LOCATION: http://192.168.1.254:5431/dyndev/uuid:105f06f1-2248-4822-f106-5f105ff1480000 SERVER: Custom/1.0 UPnP/1.0 Proc/Ver ST: upnp:rootdevice USN: uid:105f06f1-2248-4822-f106-5f105ff1480000::upnp:rootdevice
2118635	HTTP/1.1 200 OK CACHE-CONTROL: 600 EXT: – LOCATION: http://192.168.1.254:1990/WFADevice.xml SERVER: POSIX UPnP/1.0 UPnP Stack/5.110.27.2001 ST: upnp:rootdevice USN: uuid:53ea59b8-5234-0a10-ad8d-02d244fb8f10::upnp:rootdevice

(b) SSDP Sample Packets: SSDP Data

Table 5.18: SSDP Sample Packets (NetDataCoP).

The first 2 packets (PacketID 2118631 and 2118632) in Table 5.18a are SSDP request packets from *'192.168.1.64'* to *'192.168.1.254'*, this is clear from the destination port of these packets (SSDP port 1900). Likewise, packets 3 and 4 (PacketID 2118633 and 2118635) are response packets since they have SSDP port as source port.

This information can be confirmed by examining the request lines of these packets in Table 5.18b, where request line *"M-SEARCH * HTTP/1.1"* means request packet, and request line *"HTTP/1.1 200 OK"* means response packet (Section D.14).
Also, the **'ST'** (Search Target) field in the packets indicates that the discovery process between *'192.168.1.64'* and *'192.168.1.254'* concerns 'rootdevice'.

5.3 Connection Identification and Reconstruction Module

In this section, we explain how we identify and reconstruct connections. The module is able to analyze Internet Layer ICMP connections, Section 5.3.1, as well as TCP Transport Layer connections, Section 5.3.2. Additionally, and this is one of the contributions of this work, we propose a new methodology for identifying and reconstructing logical connections for UDP-based applications, Section 5.3.3.

5.3.1 ICMP Connections

As mentioned in Section 5.2.4, we focus on ICMP echo request and reply packets. Our analysis has led us to identification and reconstruction of ICMP connections based on the deciphered ICMP fields. We have illustrated the main fields of ICMP packets and clarified that ICMP echo packets (requests and replies) contain: *Identifier* and *Sequence Number* fields. We have concluded that they are responsible for matching echo requests to their respective echo replies.
We concentrated on those fields and conducted several experiments from which we were able to match ICMP echo request packets to their respective replies,

Table 5.19.

TimeStamp	SourceIP	DestinationIP	Type	Code	Identifier	Sequence Number	Data
15:24:56.603	192.168.1.71	192.168.1.66	8	0	1	112	151
15:24:56.647	192.168.1.71	192.168.1.66	8	0	1	112	151
15:24:56.893	192.168.1.66	192.168.1.71	0	0	1	112	151
15:24:57.423	192.168.1.66	192.168.1.71	0	0	1	112	151

(a) ICMP Connection Packets

Connection Information	
ConnectionID	5333
StartTime	15:24:56.603
EndTime	15:24:57.423
Duration	00:00:00.820
Protocol	1 (ICMP)
SourceIP	192.168.1.71
DestinationIP	192.168.1.66
Status	Complete
Packets	4
Data	604

(b) ICMP Identified and Reconstructed Connection

Table 5.19: ICMP Connection Example (NetDataCoP).

Table 5.19 illustrates sample ICMP connection from our captured network traffic. As we can see in Table 5.19a, the *four* illustrated packets share the same Identifier and the same Sequence Number. We identify these packets as one connection and reconstruct it out of the packets. Table 5.19b shows the reconstructed connection and its designed and developed features: duration, start time, end time, sent packets and overall exchanged data (in bytes).

A very important aspect here is the connection status. We assume that the existence of ICMP echo request packets postulates a *beginning of connection*, likewise, the occurrence of ICMP echo reply packets is treated as an *ending of connection*. As the result, we conclude the following statuses for ICMP connections, Table 5.20:

Echo Request	Echo Reply	Connection Status
✓	✓	Complete
✓	×	Incomplete E
×	✓	Incomplete B

Table 5.20: ICMP Connection Status

When *both* ICMP echo request and reply packets exist in a connection then it is a **complete** connection. When ICMP echo request packets exist *only* we regard the connection as an incomplete connection that has no ending, *i.e.*, '**Incomplete E**'. Similarly, when echo reply packets occur *only* in a connection, then we deem it as an incomplete connection with no beginning, *i.e.*, '**Incomplete B**'.

The interpretation of ICMP connections helps us to investigate the effects of ICMP Flood, Smurf, LAND, and IPSweep attack, and eventually detect their presence.

5.3.2 TCP Connections

Transport connections between hosts are usually tracked using a *Transport Address*. It consists of five fields: Source Address, Source Port, Destination Address, Destination Port, and Protocol. The protocol refers to transport protocol, in our case here it is TCP. For one direction of flow a connection is tracked by $< IP_{src}, Port_{src}, IP_{dst}, Port_{dst}, Protocol >$, and in the other direction it is tracked by $< IP_{dst}, Port_{dst}, IP_{src}, Port_{src}, Protocol >$.

We find those packets that share the same transport address (with specific Addresses and Port numbers) and group them. Within groups of packets, we track the connection establishment and termination patterns. The TCP header fields explained in Section D.7 allow unique establishment and termination patterns for TCP connections. In order to explain the patterns, we are going to use some TCP protocol fields: Sequence Number, Acknowledgment Number,

and the flag controls (only the ones which needs to be set for specific steps).

5.3.2.1 Connection Establishment

TCP Connection establishment pattern follows a three-way handshaking mechanism to initiate connections between two hosts, Figure 5.1.

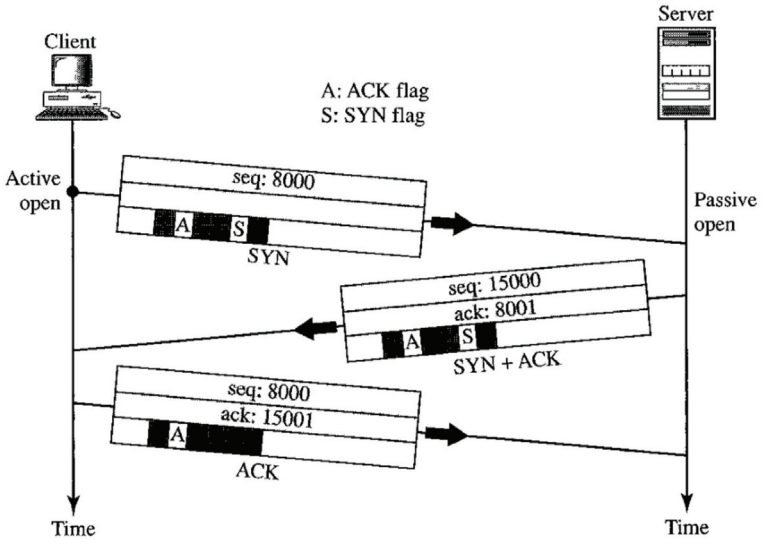


Figure 5.1: TCP Connection Establishment Pattern. [1]

When the client wishes to initiate a connection with a server, it sends a TCP packet with the *SYN flag* set, and from our experience in capturing traffic the *Acknowledgement Number* set to zero. This first packet does not carry any data and is only intended to synchronize two hosts, *i.e.*, the client is telling the server that it wants to connect to it.

If the server is not busy, it sends back a reply to the client acknowledging the request and telling it that it is ready to accept the connection. This reply packet has its *SYN and ACK flags* set and consumes one *sequence number*.

Finally, the client replies with a packet whose *ACK flag* is set and its *sequence*

number is the same as the original SYN packet sent by the client.

These three steps mark the connection establishment pattern of TCP connections. After that, a data transfer between the two hosts commences until the connection is terminated.

5.3.2.2 Connection Termination

In a three-way handshaking manner similar to Establishment pattern, the connection termination pattern starts when a host sends a FIN packet with its *FIN flag* set, Figure 5.2. Basically, any of the connection's hosts can terminate the connection, but usually the client that initiated the connection is one that terminates it. The FIN packet can contain the remaining connection's data or not. If the packet is empty (does not hold any data) it consumes one sequence number.

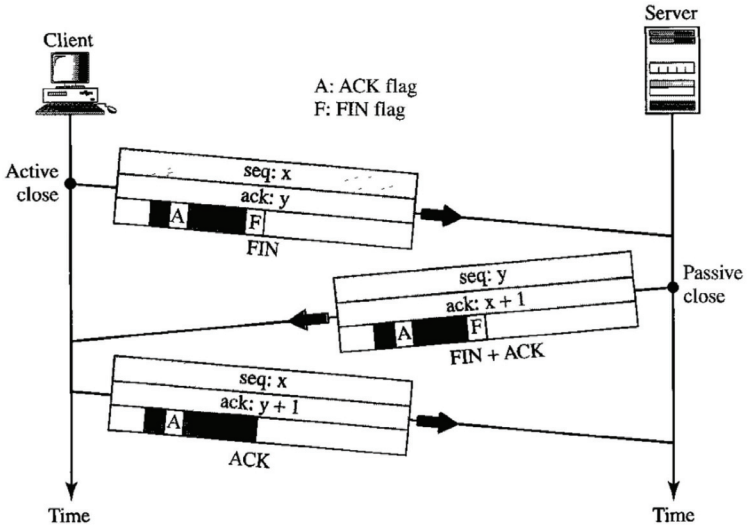


Figure 5.2: TCP Connection Termination Pattern. [1]

The server responds with packet whose ACK and FIN flags are set acknowledging the received FIN packet. This packet might contain the last piece

of connection data from the server, if it does not then it consumes one sequence number. Finally, the client replies with an ACK packet to confirm his receiving of the server's FIN packet. This packet is the last packet of the termination pattern and consequently the last packet of the connection.

There are some cases in which the connection is not terminated in this way. When the server receiving the connection initiation request is busy, or some error happened that prevents it from accepting the connection, the server replies to the connection request with a RST packet (reset), i.e., a packet whose RST flag is set. In essence, the server is refusing the connection and informing the client that it is not ready to accept it. Thus, this can also be regarded as a connection termination pattern.

In the same manner as ICMP connections, Figure 5.3 illustrates identified and reconstructed TCP connection sample from our captured network traffic.

Based on the presented steps of establishing and terminating TCP connections, we proposed the following features describing TCP connections:

- connection duration;
- connection sent number of packets;
- connection exchanged data, in bytes;
- Internet Layer features: source and destination addresses;
- Transport Layer features: source port, destination port, and connection status.

The developed NetDataCoP is able to determine all these features.

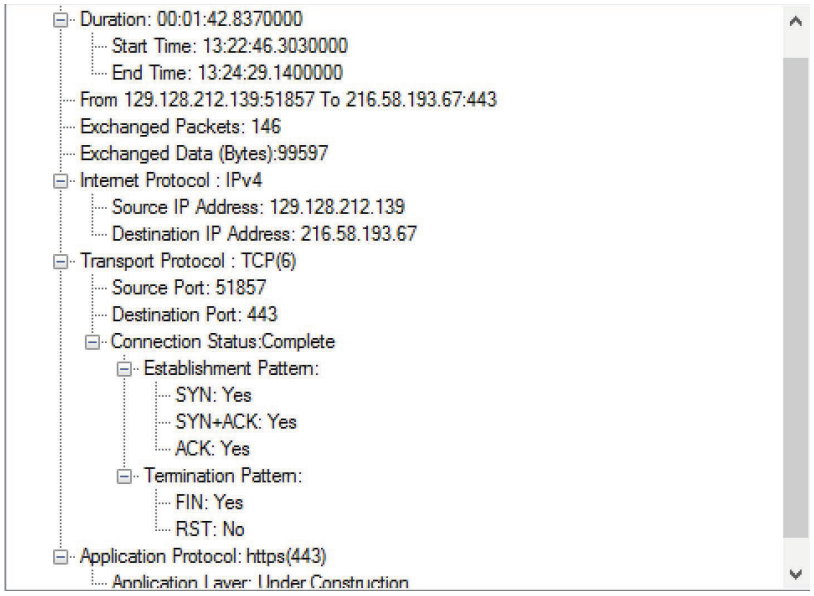


Figure 5.3: TCP Connection Example (NetDataCoP).

For TCP connections, we are able to identify more possibilities statuses than that for ICMP connections. We have 4 main flags that govern and control the connection establishment and termination patterns, namely: SYN, ACK, FIN, and RST flag. Therefore, we have multiple combinations of values of these flags, and this means multiple connection statues, Table 5.21.

ID	Establishment Pattern			Data Exchange	Termination Pattern		Connection Status
	SYN	$SYN + ACK$	ACK		FIN	RST	
1	✓	✓	✓	✓	✓	×	Complete
2	✓	✓	✓	✓	×	✓	Complete
3	✓	✓	✓	✓	×	×	IncompleteE
4	×	×	×	✓	×	✓	IncompleteB
5	×	×	×	✓	✓	×	IncompleteB
6	×	×	×	✓	×	×	Incomplete
7	✓	×	×	×	×	×	Attempted
8	✓	✓	×	×	×	×	Attempted
9	✓	×	×	×	✓	×	Attempted and Aborted
10	✓	×	×	×	×	✓	Attempted and Aborted
11	✓	✓	×	×	✓	×	Attempted and Aborted
12	✓	✓	×	×	×	✓	Attempted and Aborted

Table 5.21: TCP Connection Status

The presence of valid connection establishment pattern indicates the beginning of a new connection, while the existence of valid termination pattern marks the end of a connection. The word 'valid' indicates the completion of the pattern.

From Table 5.21, we regard the first 2 connection patterns as complete connections. The connection establishment pattern exists and the connection is properly terminated with a termination pattern (FIN or RST) after data exchange. We assume, the connection pattern in row 3 is 'Incomplete E', where a valid establishment pattern exist, and data has been exchanged with no termination patterns found.

Rows 4 and 5 are considered 'Incomplete B', where we find data exchange for connection with proper connection termination (using FIN or RST), but with no connection establishment pattern. Connection pattern in row 6 is considered 'Incomplete', where data exchange exists with neither establishment nor termination patterns. Connection patterns in rows 7 and 8 are regarded as 'Attempted' connections, in which an attempt was made to initiate a connection. Rows 9 – 12 are regarded as 'Attempted and Aborted', in which a connection initiation attempt was made but was faced with termination before it begins.

5.3.3 UDP Connections

Like for the case of TCP connections, we wanted to know if there are some kind of behavioural patterns that occur between UDP-based applications. We have investigated the concept of Deep Packet Inspection (DPI) also known as protocol analysis, in which we interpret packets of UDP-based application protocols to understand their behaviour. We have been successful in understanding this behaviour. Therefore, we propose a process of identification of logical

connections for UDP-based applications.

As explained for transport connections, we started tracking UDP packets and grouping them by their 5-tuple transport address. However, this approach of tracking and grouping UDP packets was insufficient due to the following shortcomings:

- In case of TCP connections, the transport address coupled with TCP header fields (Sequence Number, Acknowledgment Number, and the flag controls) provided unique distinction between groups of TCP packets to form connections. There are no similar fields in UDP headers that can do this job.
- Re-occurring groups of UDP packets sharing the same transport address within data results in non-unique distinction between groups of UDP packets.

To deal with these shortcomings we need additional information to use besides a transport address. This information is available within the headers of UDP-based applications. As a case study for this research, we selected the following UDP-based applications:

- Domain Name System protocol (DNS)
- NetBIOS Name Service (NBNS)
- NetBIOS Datagram Service (NBDS)
- Bootstrap Protocol (BOOTP)
- Link-Local Multicast Name Resolution (LLMNR)
- Simple Network Management Protocol (SNMP)

- Simple Service Discovery Protocol (SSDP)
- Real-Time Transport Protocol (RTP) and Real-Time Transport Control Protocol (RTCP)

Application Protocol designers realize the connectionless unreliable nature of UDP as well as the connection-oriented reliable nature of TCP. To enable computers to distinguish between multiple application protocols communicating on top of UDP, protocol designers tend to compensate for the connectionless nature of UDP by providing a sense of flow control in the form of a unique identification field (ID) in the header of the application protocol.

After detailed and thorough investigation of the previously mentioned UDP-based applications, we discovered a unique number inside the application header. This number can be used alongside the UDP transport address to provide uniqueness for each logical UDP-based connection. The ID fields for each protocol are as follows:

- ID field in DNS header, Section 5.2.9.
- NAME_TRN_ID field in NBNS header, Section 5.2.11.
- DGM_ID field in NetBIOS Datagram Service (NBDS) header [7].
- Transaction Identifier (xid) field in BOOTP header, Section 5.2.8.
- ID field in LLMNR header, Section 5.2.10.
- RequestID field in SNMP header, Section 5.2.12.
- Search Target (ST) field in SSDP header, Section 5.2.13
- 32 bit synchronization source identifier (SSRC) field in RTP header and 32 bit synchronization source identifier (SSRC) field in RTCP header [75][76][77].

We are going to show our methodology using an example of the most commonly used UDP-based application protocol: DNS. We are going to use sample data from our experiments that supports the mentioned shortcomings, Table 5.22 below.

Implemented		IPv4			UDP	
PacketID	Timestamp	Source IP	Destination IP	Protocol	Source Port	Destination Port
3066	12:17:20.2630000	129.128.212.139	129.128.208.6	17	50061	53
3067	12:17:20.3900000	129.128.208.6	129.128.212.139	17	53	50061
3903	12:19:20.7500000	129.128.212.139	129.128.208.6	17	50061	53
3909	12:19:21.7400000	129.128.208.6	129.128.212.139	17	53	50061
4107	12:19:51.5100000	129.128.212.139	129.128.208.6	17	50061	53
4108	12:19:51.6370000	129.128.208.6	129.128.212.139	17	53	50061

(a) DNS Sample Packets: Internet and Transport Data

PacketID	DNS					
	ID	QR	QCount	ARCount	NSCount	ARRCount
3066	10425	0	1	0	0	0
3067	10425	1	1	1	0	0
3903	21310	0	1	0	0	0
3909	21310	1	1	9	4	4
4107	46336	0	1	0	0	0
4108	46336	1	1	1	0	0

(b) DNS Sample Packets: DNS Data

PacketID	RR								
	QName	QType	QClass	RR_Name	RR_Type	RR_Class	TTL	RDLength	RData
3066	plus.google.com	28	1	–	–	–	–	–	–
3067	plus.google.com	28	1	plus.google.com	28	1	3120562176	16	2607:F8B0:4009:809:0:0:0:200E
3903	tiles.services.mozilla.com	1	1	–	–	–	–	–	–
3909	tiles.services.mozilla.com	1	1	tiles.r53-2.services.mozilla.com	1	1	1006632960	4	52.32.9.85
				.r53-2.services.mozilla.com	2	1	2426339840	25	ns-1537.awsdns-00.co.uk
				.ns-1537.awsdns-00.co.uk	1	1	3885892096	4	205.251.198.1
				.r53-2.services.mozilla.com	2	1	2426339840	19	ns-206.awsdns-25.com
				.r53-2.services.mozilla.com	2	1	2426339840	22	ns-772.awsdns-32.net
				.tiles.r53-2.services.mozilla.com	1	1	1006632960	4	52.32.238.102
				.tiles.r53-2.services.mozilla.com	1	1	1006632960	4	52.34.245.108
				.tiles.services.mozilla.com	5	1	738263040	14	tiles.r53-2.services.mozilla.com
				.tiles.r53-2.services.mozilla.com	1	1	1006632960	4	54.191.113.255
				.tiles.r53-2.services.mozilla.com	1	1	1006632960	4	52.34.249.78
				.tiles.r53-2.services.mozilla.com	1	1	1006632960	4	54.148.98.19
				.tiles.r53-2.services.mozilla.com	1	1	1006632960	4	54.148.230.222
				.tiles.r53-2.services.mozilla.com	1	1	1006632960	4	54.149.224.177
				.r53-2.services.mozilla.com	2	1	2426339840	23	ns-1507.awsdns-60.org
				.ns-206.awsdns-25.com	1	1	3885892096	4	205.251.192.206
				.ns-772.awsdns-32.net	1	1	3885892096	4	205.251.195.4
				.ns-1507.awsdns-60.org	1	1	3885892096	4	205.251.197.227
4107	www.beartracks.ualberta.ca	1	1	–	–	–	–	–	–
4108	www.beartracks.ualberta.ca	1	1	www.beartracks.ualberta.ca	1	1	856293376	4	142.244.120.201

(c) DNS Sample Packets: RR Data

Table 5.22: Logical DNS Connection Packets (NetDataCoP).

Table 5.22 illustrates our analysis of the hypothesis of logical UDP-based connections. It displays selected data of 6 DNS packets. Table 5.22a displays their respective Internet and Transport layer data while Table 5.22b and Table 5.22c display their respective Application layer data. From Table 5.22a we can conclude that the transport address for those packets is

$< 129.128.212.139, 50061, 129.128.208.6, 53, 17 >$. According to the claim of other researchers stating that transport address is enough for grouping packets, then we should regard these 6 packets as one group. But actually they are not.

Table 5.22b illustrates DNS header data for these packets. It is clear that packets 3066 and 3067 share the same ID: 10425. Also, packets 3903 and 3909 share the ID of 21310, while packets 4107 and 4108 have the same ID of 46336. Associating the ID field to the transport address of such packets we get 3 distinguishable groups, *i.e.*, 3 different DNS logical connections.

Furthermore, looking at the *QR* field, we can distinguish between DNS request and response packets within identified logical connections. We conclude that packet 3067 (row 2) is a response for packet 3066 (row 1), packet 3909 (row 4) is a response for packet 3903 (row 3), and finally packet 4108 (row 6) is a response for packet 4107 (row 5).

Table 5.22c supports our claim of logical connections. When we look at the queried names and their respective responses, and compare this finding to ID-based grouping of packets, then the picture of name resolution becomes clearer. Figures 5.4 to 5.6 illustrate our identification result of logical connections and their related constructed fields.

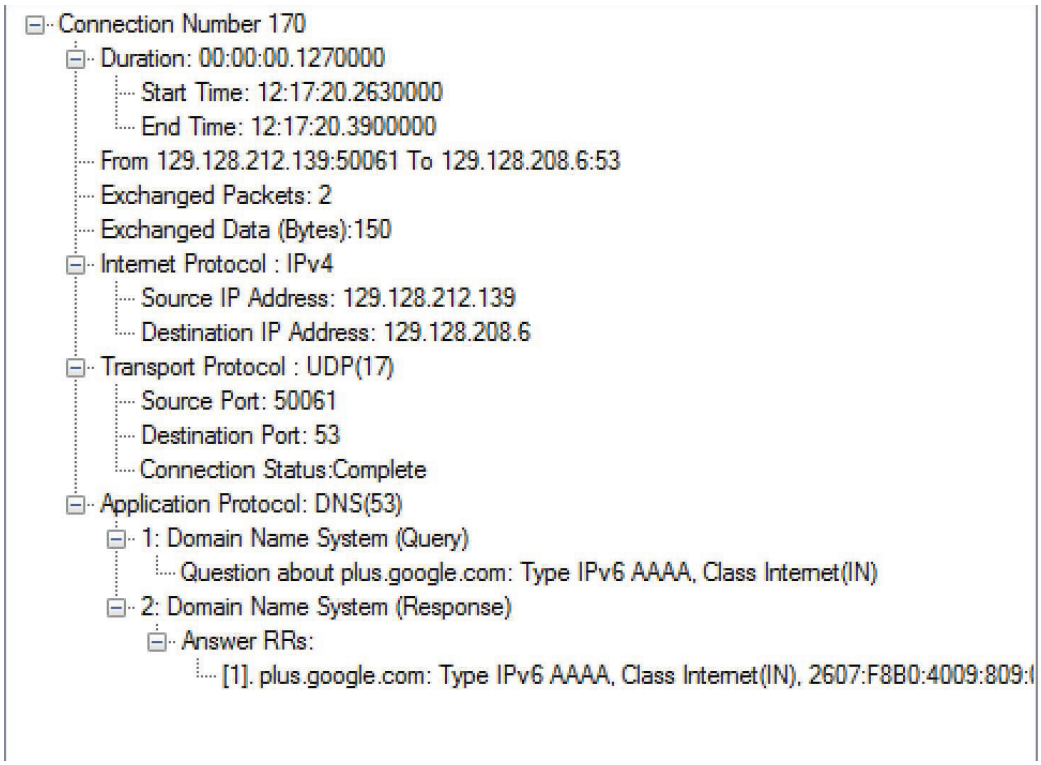


Figure 5.4: DNS Logical Connection 1 (NetDataCoP).

Figure 5.4 illustrates the DNS logical connection of packets 3066 and 3067. The **duration** of the connection is **127 millisecond**, the connection was initiated by **129.128.212.139** using source port **50061** to **129.128.208.6** via destination port **53 (DNS)**. The connection consists of **2 packets** and overall exchanged data of **150 bytes**. Its **status** is '**complete**' because it contains both DNS request and response packets. The requestor wanted to know the physical address of **plus.google.com** and it received a reply from the DNS server **2607:F8B0:4009:809:0:0:0:200E**.

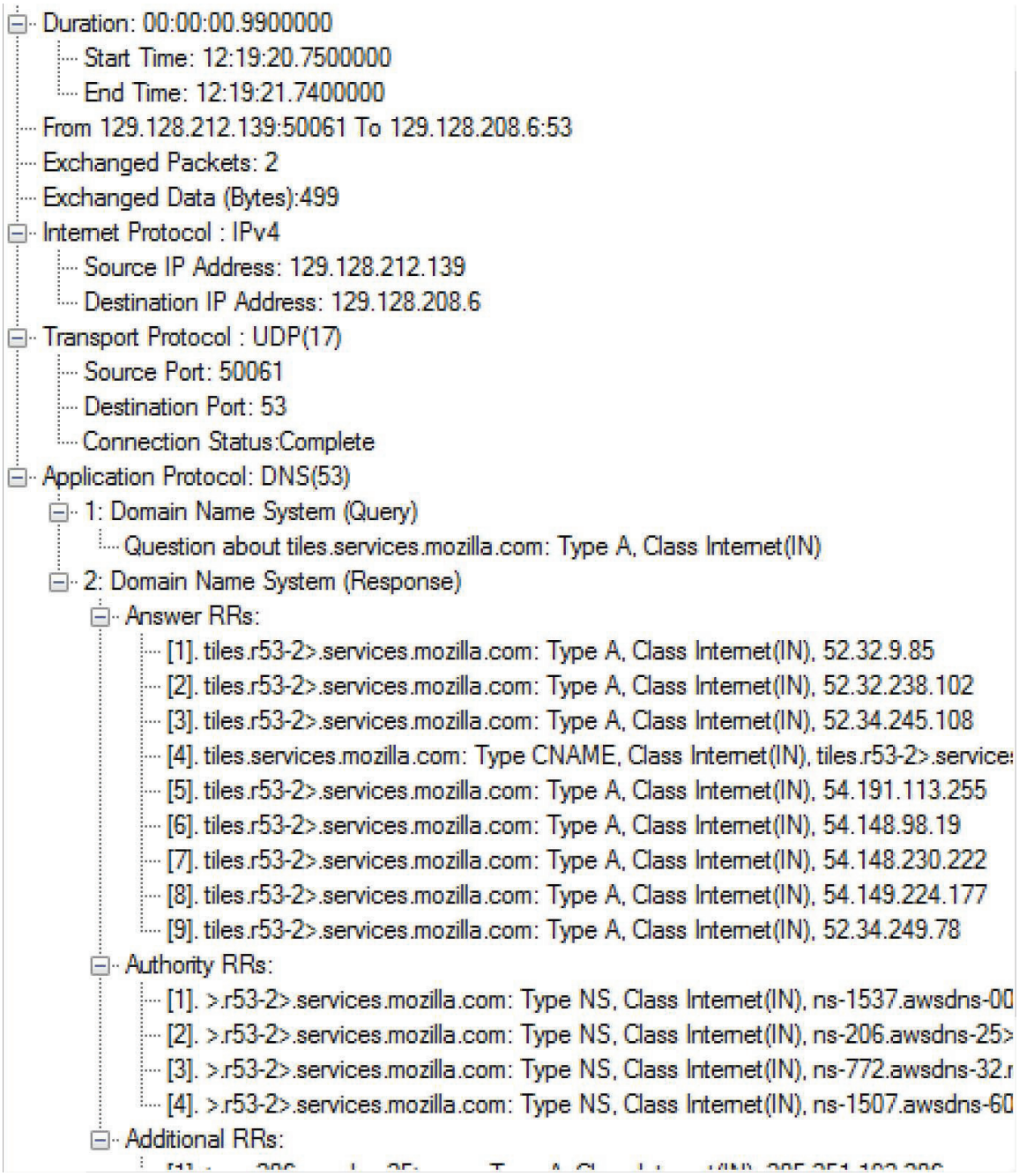


Figure 5.5: DNS Logical Connection 2 (NetDataCoP).

Likewise, DNS logical connection 2, Figure 5.5, the connection **duration** is **990 milliseconds**, between the same hosts using the same ports and same number of packets. The exchanged data is **499 bytes** because the response packet

in this connection contains more RRs, **17 RRs** to be exact: 9 RRs in the Answer section, 4 RRs in the Authority section, and 4 RRs in the Additional section. The requestor queried **tiles.services.mozilla.com** and received several information within the reply.

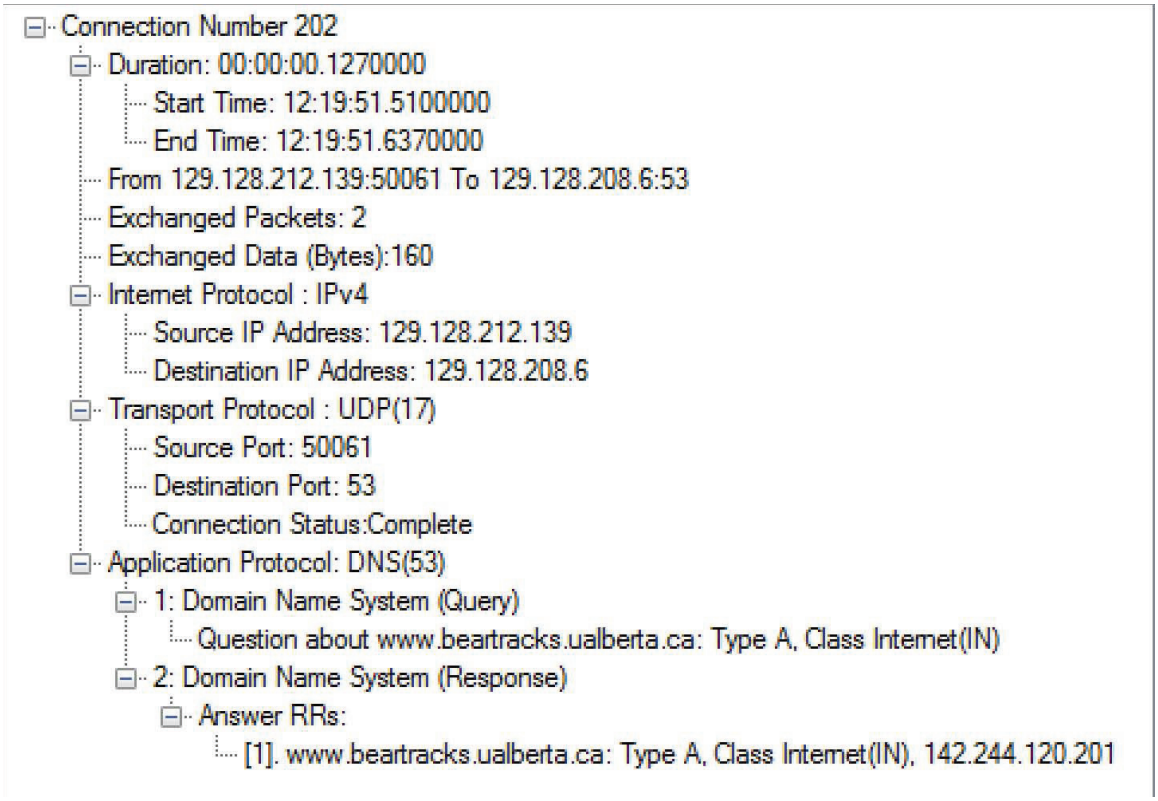


Figure 5.6: DNS Logical Connection 3 (NetDataCoP).

The third DNS logical connection, Figure 5.6, has the same fields but with different values. The connection **duration** spanned for **127 millisecond** between the same hosts using the same ports and the same number of packets. Its exchanged data is **160 bytes**. The requestor wanted to know the **IPv4 address** of **www.beartracks.ualberta.ca** and got the response of **142.244.120.201**.

The status of logical connections of UDP-based applications is the same as that of ICMP connections explained in Section 5.3.1. The different statuses are

demonstrated in Table 5.23

Request	Response	Connection Status
✓	✓	Complete
✓	×	Incomplete E
×	✓	Incomplete B

Table 5.23: UDP Connection Status

When *both* request and response packets exist in a connection then it is a **Complete** connection. Likewise, when request packets exist *only* then we regard the connection as an incomplete connection that has no ending, *i.e.*, '**Incomplete E**'. When response packets occur *only* in a connection, which is highly unlikely, then we perceive it as an incomplete connection with no beginning, *i.e.*, '**Incomplete B**'. More experiments that provide additional validation to this methodology in Appendix B.

5.4 Network Traffic Temporal Processing Module

Mainly, our temporal processing of network traffic is no different than the commonly known approach. For each minute, we calculate the values of its associated network traffic. The novelty of our approach lies within the proposed traffic features, inspired by our multi-perspective network traffic description. In this section we explain the features that we processed in a minute-wise manner for the proposed *Two-stage Hybrid Intrusion Detection System.*

The features that we use for *Global Network-level minute-wise Machine Learning-based Anomaly Detection* are the same as those we use in *Global Network-level minute-wise Machine Learning-based Signature Detection.* The difference is that in Signature ID we use multiple subsets of the minute-wise features of Anomaly ID. The features of *Local Host-level session-wise Threshold-based Anomaly Detection* are somehow different because they are

related to specific hosts. Attack Detection Modules use the network-tailored attack signature features indicated in Section 3.6.

The features of *Global Network-level minute-wise Machine Learning-based Anomaly Detection* are explained in Section 5.4.1, while those of *Local Host-level session-wise Threshold-based Anomaly Detection* are explained in Section 5.4.2.

5.4.1 Global Network-level Machine Learning-based Anomaly Detection Features

Anomaly ID deals with the detection of abnormal network traffic, *i.e.*, it distinguishes between normal and anomalous network traffic patterns. We realize the normality of network traffic from a scope-centric perspective via:

- Global Network-level minute-wise Machine Learning-based context; we discuss its features here.
- Local Host-level session-wise Threshold-based context, Section 5.4.2

Since we are dealing with the detection of anomalies using machine learning classifiers, we had to have training data. This data is used in the training phase of classifiers to build models that represent normal network behaviors. In the test phase (actual phase of detecting anomalies), network traffic is processed in the form of minute-wise data records and passed to machine learning classifiers. The classifiers classify the given data point and detects whether it is normal or anomalous, based on the pre-constructed models.

The features used in the training phase and in the actual detection phase are depicted in Table 5.24.

TCP/IP Layer	Protocol	Feature
Network Interface	ARP	ARPRequests ARPResponses ARPSourceIP ARPDestinationIP
Internet	ICMP	ICMPQueryReq ICMPQueryRes ICMPSourceIP ICMPDestinationIP ICMPConnections
	IGMP	IGMPPackets IGMPMulticastP IGMPUnicastP
	–	LAND
Transport	TCP	TCPPackets TCPConnections TCPServices
	UDP	UDPPackets UDPConnections UDPServices
Application	DNS	DNSRequests DNSResponses DNSConnections DNSMulticastD DNSUnicastD DNSNR
	NBNS	NBNSRequests NBNSResponses NBNSConnections NBNSMulticastD NBNSUnicastD NBNSNR
	LLMNR	LLMNRRequests LLMNRResponses LLMNRConnections LLMNRMulticastD LLMNRUnicastD LLMNRNR

Table 5.24: Global Network-level Anomaly Features

The features were calculated by processing Network-level traffic on a minute basis as follows:

ARPRequests Number of ARP request packets.

ARPResponses Number of ARP response packets.

ARPSourceIP Number of distinct ARP source addresses.

ARPDestinationIP Number of distinct ARP destination addresses.

ICMPQuery Number of ICMP query packets.

ICMPQueryReq Number of ICMP query request packets.

ICMPQueryRes Number of ICMP query response packets.

ICMPSourceIP Number of distinct ICMP source addresses.

ICMPDestinationIP Number of distinct ICMP destination addresses.

ICMPConnections Number of ICMP connections.

IGMPPackets Number of IGMP packets.

IGMPMulticastP Number of multicast IGMP packets.

IGMPUnicastP Number of unicast IGMP packets.

LAND A categorical field indicating a LAND attack.

TCPPackets Number of TCP packets.

TCPConnections Number of TCP connections.

TCPServices Number of TCP services.

UDPPackets Number of UDP packets.

UDPConnections Number of UDP connections.

UDPServices Number of UDP services .

DNSRequests Number of DNS request packets.

DNSResponses Number of DNS response packets.

DNSConnections Number of DNS connections.

DNSMulticastD Number of distinct DNS multicast destination addresses.

DNSUnicastD Number of distinct DNS unicast destination addresses.

DNSNR InsideSniffer Attack categorical indicator, if a DNS address to name resolution is made.

NBNSRequests Number of NBNS request packets.

NBNSResponses Number of NBNS response packets.

NBNSConnections Number of NBNS connections.

NBNSMulticastD Number of distinct NBNS multicast destination addresses.

NBNSUnicastD Number of distinct NBNS unicast destination addresses.

NBNSNR InsideSniffer Attack categorical indicator, if a NBNS address to name resolution is made.

LLMNRRequests Number of LLMNR request packets.

LLMNRResponses Number of LLMNR response packets.

LLMNRConnections Number of LLMNR connections.

LLMNRMulticastD Number of distinct LLMNR multicast destination addresses.

LLMNRUnicastD Number of distinct LLMNR unicast destination addresses.

LLMNRNR InsideSniffer Attack categorical indicator, if a LLMNR address to name resolution is made.

5.4.2 Local Host-level Features

In this context, we designed and calculated session-wise features. Unlike Global Anomaly detection, we create session-based thresholds of 27 features; Table 5.25.

During the training phase, we use the same data as in Global Anomaly ID, but here we produce session-wise host-related data records, where each data record represents the traffic of individual hosts. We thoroughly explain the concept of *Packet Capture Sessions* in the next chapter.

Out of these records we extract and use only the normal ones. For each designed feature, we search through the feature values within the extracted normal data records and we choose the maximum value. Thus, we end up having 27 thresholds that represent the maximum normal behavior of individual host, Table 5.25.

TCP/IP Layer	Protocol	Feature
Network Interface	ARP	Destinations Per Source Requests Per Destinations Requests Per Source Responses Per Sources Responses Per Destination
Internet	ICMP	Destinations Per Source Requests Per Destinations Requests Per Source Responses Per Sources Responses Per Destination
	IGMP	Multicast Packets Per Second Unicast Packets Per Second
	–	LAND Indicator
Transport	TCP	TCP Packets TCP Connections TCP Services
	UDP	UDP Packets UDP Connections UDP Services
Application	DNS NBNS LLMNR	DNS NR NBNS NR LLMNR NR

Table 5.25: Local Host-level Anomaly Features

During the detection phase, once a minute is classified as anomalous, we extract the traffic of each of its sources and perform a simple comparison between the minute calculated host features and the pre-constructed normal thresholds.

Chapter 6

Network Traffic Generation

The performed analysis of attacks and existing intrusion detection datasets has led us to the conclusion that we need to generate new datasets that represent network traffic using an extended set of features. In particular, a network traffic data should be described using our proposed multi-perspective set of features. Thus, we have decided to build our own environment for data generation purposes. This has allowed us to design such an environment 'from scratch' and equip it with methods for generating network traffic representing different normal, anomalous, and attack scenarios.

In this chapter, we explain the main aspects of this environment: network traffic generation, and data capturing, Section 6.2; as well as the generated data, Section 6.3.

6.1 Network Environment

We have designed and implemented a network of eight devices to perform intrusion detection experiments. The devices are connected to a main switch with port-mirroring capabilities, which is consequently connected to a router that connects the network to the Internet. A mirroring option of the switch is

used for monitoring purposes; it copies all network traffic passing through the switch to a mirror port. Our implemented IDS is connected to this port. In other words, the switch delivers a copy of the whole network traffic to a network card of IDS.

In the normal capture mode, network cards monitor network traffic data and pick up data packets by investigating the source and destination addresses of each packet. If one of these addresses is the address of a given network card, the card 'takes' this packet from the traffic.

In order to allow the IDS to capture all traffic, also the one not directed towards its network card, the IDS's network card capture mode has to be set to the'*promiscuous*' mode. Thus, all the traffic in a network can be captured by the IDS's network card. A software system able to capture and process the captured network traffic data has been presented in Chapter 5.

We use this built test-bed network environment to perform various experiments, some of which include capturing a normal network traffic data, while others include capturing intrusive network traffic data. Additionally, some experiments contain intrusive data embedded within normal traffic.

6.2 Network Traffic Generation

In this section we explain how we generate network traffic data representing normal and attack network conditions. Also, we provide an explanation of a concept of packet capture sessions.

6.2.1 Packet Capture Sessions

In our context of capturing network traffic, we use a concept of '*Packet Capture Session*'. A capture session is a name given to a single session of our capturing software (Chapter 5) performed over a user-defined duration. We categorize packet capture sessions into the following classes:

- Normal Packet Capture Sessions that contain a normal, *i.e.*, attack free, network traffic;
- Attack Packet Capture Sessions that contain an attack traffic only;
- Overlay Packet Capture Sessions that contain both attack and normal traffic; this type of sessions are the most realistic intrusion scenarios due to the fact that attackers try to hide or conceal their intrusive activities within a normal network traffic making attacks difficult to detect.

Our intention of adopting the concept of *Session* is to have time-independent packet capture intervals. Sessions allow us to focus on a specific type of network traffic pattern. This provides us with the ability to investigate different network traffic patterns within several networking scenarios without binding ourselves to a specific time duration. Thus, most of our Normal Packet Capture Sessions would last for hours at a time, contrarily, some Attack Packet Capture Sessions last for several minutes or seconds, while Overlay Packet Capture Sessions are relatively long sessions that include both normal and attack traffic. As the result, we have sessions that contain up to 73,257 packets, and others containing as low as 31 packets. Table 6.1 demonstrates session-based network traffic statistics.

Overall Sessions	Criteria	Session Types		
		Normal (52)	Attack (77)	Overlay (62)
191	Packets: 2,090,820	1,219,604	138,658	732,558
	Bytes: 623.2 Mbyte	427.6 Mbyte	10.6 Mbyte	185 Mbyte
	Durations 88:02 Hr 5282 min	63:13Hr 3793 min	4:09 Hr 249 min	20:40 Hr 1240 min

Table 6.1: Statistics of Network Traffic Sessions

6.2.2 Network Traffic Data Sets

In the presented work, any network traffic data collected with different *'Packet Capture Sessions'* is divided into three different datasets: Training, Testing, and Evaluation, Table 6.2.

Dataset	Sessions	Packets	Duration	
			Hr	Min
Training Dataset	52 Normal	1,219,604	63:13	3793
	77 Attack	138658	4:09	249
	50 Overlay	505377	12:43	763
Sub Total:	179	1,863,639	80:05	4805
Testing Dataset	5 Overlay	69,761	3:37	217
Evaluation Dataset	7 Overlay	157,420	4:20	260
Overall:	191	2,090,820	88:02	5282

Table 6.2: Statistics of Collected Network Traffic

Each dataset can be perceived as a table. The table's rows (data records) represent data objects, while the columns represent attributes (or features) that describe a given data object.

For example, in Table 6.2, the Training dataset includes data objects that have been generated during normal, attack, and overlay sessions, while the Testing and Evaluation datasets include only data objects from overlay sessions.

6.2.3 Data Generation

A process of generation of network traffic data focuses on normal and attack traffic. For the case of all intrusion detection research we investigated, normal traffic is generated using one of two approaches:

- application of traffic generation tools producing traffic inside a network;
- collection of real life traffic generated by hosts of a network.

The **first** approach based on generation tools has the following deficiencies:

1. The tools suppose to mimic a traffic of the entire network, including unicast and multicast modes; however, they are not able to fully simulate a normal traffic, they can only imitate limited network traffic patterns that could be quite different when compared to real traffic patterns; there is not much information about distribution of different type of packets in a traffic data;
2. Traffic patterns generated by these tools is well known, thus it is easy to distinguish the patterns and separate them from an attack traffic;
3. Traffic generated using these tools does not contain a full information about a real network traffic; for example, there is no information related to lower and upper layers of a protocol stack;
4. The tools generate error-free traffic that is not a true representation of a real-life network traffic.

The **second** approach of generating network traffic uses actual hosts and has only *'one'* disadvantage: a problem of confidentiality. After capturing the traffic, there is an ethical obligation to protect privacy of the captured data. This situation is clearly observed when the captured data is formatted as Intrusion Detection dataset and publicly provided to researchers in the field.

As the result, we have selected the second approach for generation of datasets. In our case, we have performed multiple browsing activities on hosts of our implemented network. In such a way, we have captured a normal traffic. For an attack traffic representing a selected set of seven attacks, the following tools have been used to simulate them:

- Hyenae [78] is a platform independent tool used to generate DOS attacks; we have used it to simulate:
  1. ICMP Flood attack;
  2. IGMP Flood attack;
  3. LAND attack;
- Ostinato [79] is a traffic generator tool we have used to simulate Smurf attack;
- Solarwinds Ping Sweep [80], Angry IP Scanner [81], and Advanced IP Scanner [82] are tools for scanning for connected hosts, we use them to execute IPSweep attack;
- Solarwinds DNS Audit [83], Solarwinds DNS Who Is Resolver [84], and NetBScanner [85] are tools for scanning for computer names of a given IP Address range, we use them to execute InsideSniffer attack;
- Finally, Nmap ("Network Mapper") [86] is a scanning tool used to scan all ports of a computer system, we use it to execute PortScan attack.

These tools have been used not only with default settings, but also with various settings allowing us to generate multiple patterns for each attack.

6.3 Proposed Intrusion Detection Benchmark Dataset

As it has been explained, the anomalous and attack network traffic data we process includes feature values calculated over a one-minute interval. Also, we have mentioned earlier (Section 6.2.2) that three datasets are used in our

research. We make this data available to public to enable other researchers of Intrusion Detection system to use them as a benchmark dataset.

The proposed benchmark dataset has three aspects that do not exist in any other Intrusion Detection dataset.

The **first** aspect is related to data records. They represent a network traffic aggregated over a period of one minute. No other Intrusion Detection dataset provides such a traffic representation. Even if there are some Intrusion Detection research projects that process a network traffic represented temporally, the researchers neither provide a corresponding dataset nor they use new features, they abided by the features proposed in KDD. Most of the similar Intrusion Detection datasets represent data records as in a form of information about connections or packets, not time-wise. The **second** novel aspect is a set of features describing data traffic. They are network independent as they do not relate to a specific network topology. Any researcher can generate his own traffic, captures it, processes it, and constructs the features we proposed. The features are designed based on a comprehensive multi-perspective description of network traffic. Though the proposed benchmark contains some features which are similar to KDD and NSL-KDD, it also contains new features, namely: Network Interface features, and application layer features. No other Intrusion Detection dataset contains these features, even KDD and NSL-KDD datasets.

One might argue that the *'content'* features of KDD and NSL-KDD are application layer features. But, they are based on the network used to generate KDD. KDD contains *'temporal'* features, while our benchmark includes temporal information about traffic. Compared to KDD, the only thing that is missing in our benchmark dataset is *'host'* features. As we previously mentioned, our benchmark represents overall traffic of a network, not a traffic of individual hosts. The **third and final** aspect is realism concerning attacks. We process

network traffic in a temporal manner. In real life networking scenarios, multiple attacks can occur simultaneously, thus multiple attacks can be present within one minute, *i.e.*, within one data record. Unlike any other Intrusion Detection dataset that provides one label for each data record, our dataset provides multiple attack labels for each individual data record.

Our proposed benchmark dataset can support both: Anomaly Detection and Signature Detection. It is an aggregation of all three datasets (Training, Testing, and Evaluation). It collectively contains 5282 data records: 4818 normal data records, and 464 anomalous data records. A single label designates normal and anomalous records, there are also seven labels – one corresponding to each attack – that signifies attack classes. All of these labels can be used in classification. Table 6.3 represents the distribution of attacks within 464 anomalous data records.

Attack	Instances
ICMP Flood	121
IGMP Flood	49
LAND	56
Smurf	22
IPSweep	150
InsideSniffer	124
PortScan	11

Table 6.3: The Distribution of Attacks within Benchmark Dataset

The proposed benchmark dataset is composed of 37 features that describe each one-minute data record. The features span over four TCP/IP layers, and are related to eight protocols, Table 6.4.

TCP/IP Layer	Protocol	Feature
Network Interface	ARP	ARPRequests ARPResponses ARPSourceIP ARPDestinationIP
Internet	ICMP	ICMPQueryReq ICMPQueryRes ICMPSourceIP ICMPDestinationIP ICMPConnections
	IGMP	IGMPPackets IGMPMulticastP IGMPUnicastP
	–	LAND
Transport	TCP	TCPPackets TCPConnections TCPServices
	UDP	UDPPackets UDPConnections UDPServices
Application	DNS	DNSRequests DNSResponses DNSConnections DNSMulticastD DNSUnicastD DNSNR
	NBNS	NBNSRequests NBNSResponses NBNSConnections NBNSMulticastD NBNSUnicastD NBNSNR
	LLMNR	LLMNRRequests LLMNRResponses LLMNRConnections LLMNRMulticastD LLMNRUnicastD LLMNRNR

Table 6.4: Features of Benchmark Dataset

Chapter 7

Anomaly Intrusion Detection

Identification of anomalous network traffic can be done using different classifiers. They are built with Machine Learning (ML) techniques and available network traffic data. Besides, we used, designed, implemented, and utilized threshold-based techniques. Multiple traffic models can be used to classify a network traffic. In general, models are of different quality. Their prediction capabilities differ depending on the applied construction method. Additionally, in most cases a binary YES/NO classification is obtained, but we tried to depart from that crisp classification to probabilistic one. Our proposed Anomaly Detection context investigates minute-wise network traffic from a scope-centric perspective; global network-level scope (next section) and local host-level scope (Section 7.2).

7.1 Global network-level Machine Learning-based Anomaly Detection

Our proposed Global network-level Machine Learning-based minute-wise Anomaly Detection approach is composed of five classifiers. Each classifier

provides a classification which is associated with a belief mass that is perceived as a level of confidence in the obtained classification. Belief masses are determined based on prediction quality of classifiers. The classification results and belief masses of individual classifiers are used to obtain a final classification result. A degree of confidence in this result is determined.

7.1.1 Implementation

We propose a solution that uses elements of evidence theory, the TBM in particular, to integrate classification results from multiple classifiers and to determine a level of confidence in the aggregated result. We postulate, that application of multiple classifiers increases prediction accuracy. Each classifier is constructed using a different ML method which means that each classifier '*focuses*' on different aspects of data. Therefore, aggregation of results, or aspects, obtained from multiple classifiers should lead to better classification results.

An aggregation process is realized via the usage of evidence theory. We treat each classification result as an evidence supporting prediction of a normal or anomalous state of a network. A level of trust in that evidence is directly associated with the quality of prediction of a given classifier. Our proposed approach is illustrated in Figure 7.1. As it can be seen, each classifier is associated with a belief mass (bbm) that represents a level of confidence in its prediction ability. In general, different classification performance measures can be used as a confidence level: accuracy of prediction, precision, recall, sensitivity, or specificity.

When a given classifier makes a prediction, this predication is '*weighted*' with the classifier's bbm. Each single prediction of each classifier provides a bbm. The obtained belief masses are used to determine a final confidence level in

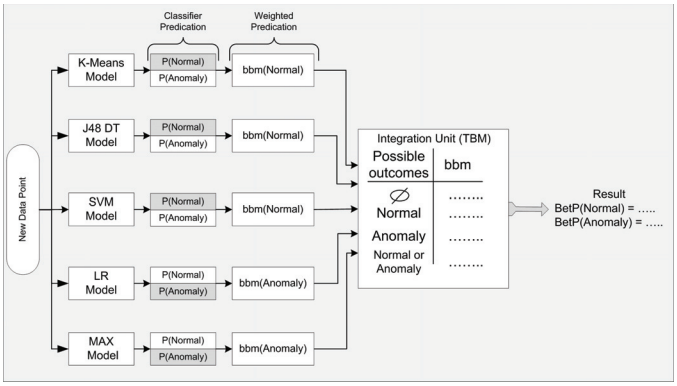


Figure 7.1: Architecture of Proposed Global Network-level Machine Learning-based Anomaly Detection Approach

the classification outcome. In the approach presented here, we use the TBM to calculate probability of each outcome. However, other methods of evidence theory or belief revision can also be used.

7.1.1.1 Determining bbm Values

The key component of the proposed approach is a process of determining values of bbm's that are associated with each classification result. Our idea is to '*link*' belief masses – treated as degrees of trust in predictions – with levels of quality of classifiers. We use *specificity* as a measure of prediction quality. Specificity represents a percentage of anomalous network traffic that has been recognized as such, *i.e.*, how good a classifier is in recognizing anomalous conditions.

We propose a *three-stage* process for estimating prediction abilities of classifiers and representing them as belief masses. At the **first** stage, belief masses are determined based on the classification results obtained during a training process, we named them bbm^{TR}; at the **second** stage, testing data set is used and new values of masses, bbm^{TS}, are obtained; at the **third** stage, both training bbm^{TR}'s and testing bbm^{TS}'s are combined to determine belief masses, bbm^{UPd}'s,

that are more realistic estimations of the classifiers' performance. For the third stage, we investigate different options of calculating bbm^{UPd}'s. The three options are:

Option A: using Bernoulli's Combination Rule;

Option B: taking only bbm^{TR};

Option C: averaging bbm^{TR}'s and bbm^{TS}'s.

7.1.1.2 Integration of Classification Results

The architecture of the system, Figure 7.1, shows that its output is the result of the TBM. The module works in the following way. When a new data point representing network traffic enters the system, it is sent to all classifiers. The output of each classifier is weighted with its bbm_C^{update} value (later, C is replaced with an abbreviation of a specific classifier).

As we have mentioned earlier, each mass represents a degree of confidence in the classifier's classification outcome, this point resembles the credal level of TBM. The classifiers' bbm_C^{update} values are input to the TBM. There are two alternatives of TBM's $BetFrame$: Normal Network Status, Anomalous Network Status (Equation 2.13). Once the TBM processes bbm_C^{update}'s, each alternative is associated with a pignistic probability $BetP$ (Section 2.1.3). The one with the highest value, out of *BetP(N)* and *BetP(A)*, is treated as the output of the system.

7.1.1.3 Construction of Classifiers

Classifiers used in this work are built using four Machine Learning (ML) methods, while the fifth classifier is constructed using a simple statistical analysis of the

data. The ML based classifiers are: *K-means*, *J48 Decision Tree*, *Support Vector Machine*, and *Logistic Regression*. All of them have been built using WEKA. The training data contains 4407 normal and 398 anomalous data records (points); while the testing data contains 153 normal and 64 anomalous data records (Section 6.2.2).

Machine Learning based Classifiers The performance of the classifiers for both training and testing data are presented in Table 7.1. The table shows specificity values obtained for the training data – bbm^{TR} in the first row, and specificity values for the testing data – bbm^{TS} in the second row. A quick look at the content indicates that *K-means* classifier has the worst performance, while *DT* has the highest values of performance measures.

Threshold-based Classifier In addition to ML classifiers, we have developed a statistical based model. Among a number of options and tries the best classification has been obtained for a '*maximum-based model*'. The classifier, hereafter called MAX-based classifier, is a feature-based model that is a collection of threshold values that are maximum values of features over a period of one minute.

As in the case of ML-based classifiers, we have evaluated the performance of MAX-based classifier using the same training and testing data sets, Table 7.1. As it can be seen, the performance of this classifier is very good, if not the best. Only the values obtained for *DT* classifier are comparable.

specificity for:	K-Means	DT	SVM	LR	MAX
training data – bbm_{TR}	0.588	0.834	0.874	0.731	0.839
testing data – bbm_{TS}	0.406	0.906	0.625	0.891	1.000

Table 7.1: Specificity Values – bbm's – for Training and Testing Data

7.1.2 Evaluation

The evaluation of the proposed approach is presented using two aspects of the approach's performance: overall classification, and analysis of misclassified cases. The misclassified cases are defined as the ones for which the TBM's $BetP$ values for each alternative, *i.e.*, *BetP(N)* and *BetP(A)* exceed a threshold. Here, the value of threshold is set to 0.1.

The evaluation dataset contains 182 normal and 78 anomalous data records (Section 6.2.2). The classification results of each individual classifier for this data are presented in Table 7.2 (subscript T represents the TRUE – original – network status, while P means the PREDICTED status). The values of performance measures, such as accuracy (AC), specificity (SP), and sensitivity (SN) are included. As we can see, *DT* seems to provide the best performance, while *K-Means* the worst, meanwhile, it is a bit surprising to see an average performance of *SVM*. Let us take a look at the performance of the proposed approach.

	K-Means		**DT**		**SVM**		**LR**		**MAX**	
	N_P	A_P	N_P	A_P	N_P	A_P	N_P	A_P	N_P	A_P
N_T	53	129	182	0	181	1	166	16	179	3
A_T	12	66	3	75	37	41	8	70	2	76
	AC:	45.8	AC:	98.8	AC:	85.4	AC:	90.8	AC:	98.1
	SP:	84.6	SP:	96.2	SP:	52.6	SP:	89.7	SP:	97.4
	SN:	29.1	SN:	100.0	SN:	99.45	SN:	91.2	SN:	98.4

Table 7.2: Performance of Classifiers for Evaluation Data

7.1.2.1 System Performance

As it has been explained in Section 7.1.1.1, we use three options for determining values of bbm^{UPd}. We show the performance of the system for each of them.

Option A $bbm^{UPd=Bern}$**:** This approach uses the Bernoulli's Rule of Combination that is a special case of Dempster-Shafer Rule of Combination. Table 7.3

illustrates belief masses calculated based on bbm^{TR} and bbm^{TS}. The performance of the proposed approach with $bbm^{UPd=Bern}$ for the evaluation dataset is illustrated in Table 7.4. The values of performance measures are: accuracy – 96.92%, specificity – 94.87%, sensitivity – 97.80%.

Option B $bbm^{UP=Train}$**:** Table 7.3 contains belief masses associated with each classifier. These masses are the ones obtained for the training dataset – bbm^{TR}. The proposed system with masses $bbm^{UPd=Train}$ provides the results shown in Table 7.4. The classification results are: accuracy of 98.46%, while specificity and sensitivity are 96.15% and 99.45%, respectively.

Option C $bbm^{UPd=Avg}$**:** Another approach for determining the bbm^{UPd}'s used here is a simple average of bbm^{TR} and bbm^{TS}, Table 7.3. The results of classification are illustrated in Table 7.4. The values of performance measures are: accuracy of 99.23%, additionally specificity equals 97.43%, and sensitivity is 100.00%.

The presented results provide a very interesting interpretation of investigated options used to combine individual bbm's. It seems that the simple averaging leads to the best prediction results, while the Bernoulli's rule of combination to the worst. A quick look at the values of $bbm^{UPd=\cdots}$s, Table 7.3, shows that lower values of bbm's give better results. It seems the Bernoulli's rule substantially increases values of bbm's and such situation leads to a number of situations where TBM struggles to provide any (meaningful) values of pigmistic probabilities (see next section).

	K-Means	DT	SVM	LR	MAX
Option A: $bbm^{UPd=Bern}$	0.76	0.98	0.95	0.97	1.00
Option B: $bbm^{UPd=Train}$	0.59	0.83	0.87	0.72	0.84
Option C: $bbm^{UPd=Avg}$	0.50	0.87	0.75	0.80	0.92

Table 7.3: bbm^{UPd} – Updated Belief Masses

	$bbm^{UPd=Bern}$		$bbm^{UPd=Train}$		$bbm^{UPd=Avg}$	
	N_P	A_P	N_P	A_P	N_P	A_P
N_T	178	4	181	1	182	0
A_T	4	74	3	75	1	77
	AC:	96.92	AC:	98.46	AC:	99.6
	SN:	97.80	SN:	99.45	SN:	100
	SP:	94.87	SP:	96.15	SP:	98.7

Table 7.4: Classification Results – the Approach with:

A quick look at the results indicates that *Option C*, $bbm^{UPd=Avg}$, produces the best classification results.

7.1.2.2 Analysis of Misclassified Cases

The proposed approach has resulted in some misclassifications, Table 7.5, Table 7.6, and Table 7.7. For the case with $bbm^{UPd=Bern}$, the approach has four false positives (FP's) and four false negatives (FN's), with $bbm^{UPd=Train}$ it has three FP's and and one FN, while the approach with $bbm^{UPd=Avg}$ only one FP's and no FN.

Let us take a closer look at the presented cases. For $bbm^{UPd=Bern}$, Table 7.5, we include two examples that are representative of a total of eight misclassifications. Firstly, let us look at the data point **org-N**: the classifiers *K-Means*, *DT*, and *SVM* classify the point properly. However, both *LR* and *MAX* have misclassified it. Both of them have large values of bbm^{UPd}'s that are 'balanced' by other classifiers – this results in both **BetP(N)** and **BetP(A)** equal to 0.

		K-Means	DT	SVM	LR	MAX	**BetP(N)**	**BetP(A)**
org-N	pred-N	0.76	0.98	0.95			0	0
	pred-A				*0.97*	*1*		
org-A	pred-N		0.98			1	0	0
	pred-A	*0.76*		*0.95*	*0.97*			

Table 7.5: Sample of Misclassified Data Points: System with $bbm^{UPd=Bern}$

Secondly, the data point **org-A**. As in the previous case, two classifiers – *DT* and *MAX* – have classified this point as $Normal$, and again their large values

of bbm^{UPd}'s are balanced by bbm^{UPd}'s of other classifiers and both **BetP(N)** and **BetP(A)** are again equal to 0. In those cases, we can interpret such a behaviour of the network as suspicious, and that would mean a need for further investigation.

When the approach is implemented using $bbm^{UPd=Train}$, we have a total of four misclassifications. All cases are shown in Table 7.6.

		K-Means	DT	SVM	LR	MAX	**BetP(N)**	**BetP(A)**
org-A	pred-N pred-A	0.59	*0.83*	0.87	0.72	*0.84*	0.642	0.358
org-A	pred-N pred-A	0.59	0.83	0.87	*0.72*	*0.84*	0.830	0.170
org-N	pred-N pred-A	*0.59*	0.83	*0.87*	0.72	0.84	0.357	0.643
org-A	pred-N pred-A	*0.59*	*0.83*	0.87	*0.72*	0.84	0.500	0.500

Table 7.6: Misclassified Data Points: System with $bbm^{UPd=Train}$

The first two cases are for the point **org-A**. Here, we have a set of scenarios with three classifiers predicting data points as $Normal$, while two as $Anomalous$. The strong confidence of the two classifiers has created a situation where TBM provides non-zero values for both **BetP(N)** and **BetP(A)**.

For the third case – point **org-N**, the strongest classifier *SVM*, with a help of *K-Means*, is able to '*force*' the TBM to assign a higher value to **BetP(A)**. The last case – point **org-A**, shows a '*draw*' between **BetP(N)** and **BetP(A)** – a combination of strength, *i.e.*, bbm^{UPd} values, and a number of classifiers identifying each alternative creates a balance between **BetP()** values.

The best performance has been obtained for the approach with $bbm^{UPd=Avg}$. It has only misclassified one **org-A** data point, Table 7.7. The two classifiers, *LR* and *MAX* have been '*against*' the other three. The highest value of bbm^{UPd} for *MAX* has counterweighted the bbm^{UPd}'s of other classifiers, and this has resulted in nearly equal probability values: **BetP(N) = 0.5031 and BetP(A) = 0.4969**.

		K-Means	DT	SVM	LR	MAX	**BetP(N)**	**BetP(A)**
org-A	pred-N				0.80	0.92	0.5031	0.4969
	pred-A	*0.50*	*0.87*	*0.75*				

Table 7.7: Misclassified Data Points: System with $bbm^{UPd=Avg}$

To Summarize, our proposed Global network-level Machine Learning-based minute-wise anomaly Intrusion Detection approach presents description and performance results for detecting anomalous network traffic. The approach aggregates classification results of multiple classifiers using elements of evidence theory and provides better results than that of a single classifier. Different methods of combining belief masses, representing confidence levels in classifiers obtained with different datasets, are investigated.

Based on the obtained classification results, we can state that a simple averaging of belief masses obtained using different datasets provides the best results. It seems that performance of the approach is at its best when confidence levels – *i.e.*, belief masses – associated with individual classifiers are moderate, Table 7.3 – row three.

The application of evidence theory gives an opportunity to identify network traffic that shows symptoms of suspicious network behaviour. Even in all the cases of misclassification, our proposed evidence theory based approach has provided, up to some degree, indications that the network traffic is suspicious.

7.1.3 Illustrative Example

Having explained our proposed approach and its related evaluation, we implemented it using $bbm^{UPd=Avg}$ and in this section we want to demonstrate an example of how we detect anomalies within network traffic. Table 7.8 illustrates a minute network traffic data record.

TCP/IP Layer	Protocol	Feature	Minute
Network Interface	ARP	ARPRequests	442
		ARPResponses	3
		ARPSourceIP	4
		ARPDestinationIP	150
Internet	ICMP	ICMPQueryReq	52
		ICMPQueryRes	0
		ICMPSourceIP	1
		ICMPDestinationIP	2
		ICMPConnections	47
	IGMP	IGMPPackets	10
		IGMPMulticastP	10
		IGMPUnicastP	0
	–	LAND	0
Transport	TCP	TCPPackets	140
		TCPConnections	12
		TCPServices	3
	UDP	UDPPackets	97
		UDPConnections	19
		UDPServices	2
Application	DNS	DNSRequests	0
		DNSResponses	0
		DNSConnections	0
		DNSMulticastD	0
		DNSUnicastD	0
		DNSNR	0
	NBNS	NBNSRequests	0
		NBNSResponses	0
		NBNSConnections	0
		NBNSMulticastD	0
		NBNSUnicastD	0
		NBNSNR	0
	LLMNR	LLMNRRequests	0
		LLMNRResponses	0
		LLMNRConnections	0
		LLMNRMulticastD	0
		LLMNRUnicastD	0
		LLMNRNR	0

Table 7.8: Minute Network Traffic Data

This minute network traffic data is passed to every classifier, which in turn processes the data and gives a prediction whether it is normal or anomaly. The predictions of classifiers are weighted using $bbm^{UPd=Avg}$ and passed to TBM as inputs. TBM takes these weighted predictions and computes a probabilistic outcome.

Table 7.9 illustrates the classification result of data given in Table 7.8. The first column indicates the original class of the minute data record. The second column indicates the prediction of classifiers; **pred-N** indicates that the classifier classifies the data as normal, and **pred-A** signifies that the classifiers classifies the data as anomaly.

Class	Prediction	K-Means	DT	SVM	LR	MAX	**BetP(A)**	**BetP(N)**
Anomaly	pred-N	–	–	0.75	–	–	0.9962	0.0038
	pred-A	*0.5*	*0.87*	–	*0.8*	*0.92*		

Table 7.9: Results of Classifiers

The next 5 columns illustrate the weighted predictions of the classifiers, whether they predict the data as normal (row 1) or anomaly (row 2). The last two columns represent the classification probabilistic outcome of TBM; BetP(A) and BetP(N) respectively. The result of the TBM indicate the data has been correctly identified as anomaly with 99.62% probability.

7.2 Local Host-level Threshold-based Anomaly Detection

The second part of our Anomaly ID context involves the detection of anomalies in the traffic of individual hosts, as explained in our proposed system (Chapter 4). This Local Host-level detection approach is not applied (invoked) unless the Global Network-level approach detects an anomaly. Local Host-level detection has 2 main advantages:

Gather Information about detected Anomaly: At this point, an anomaly is detected – surely – but we do not know anything about it. We do not know, for example, which protocol was used to create this anomaly? does this anomaly relate to packets or connections? So, we need this step to gather quantifiable information about the detected anomaly.

Find the sources of Anomaly: We postulate, that the global indication of anomaly is the result of one or more local anomalies that are executed by one or more hosts simultaneously. The existence of an anomaly is the

collective aggregation of one or more host anomalies. Thus, we need to know the anomalous hosts.

In Section 3.5 and Appendix A we represent the proposed network anomaly profile. We explained how we designed its features and how their values are calculated. We demonstrated our proposed Host-level features in Section 5.4.2, we notice that these features are the host-scope features of our proposed network profile. We use these features to detect Local Host-level anomalies. Table 7.10 represents these features and their respective threshold values.

TCP/IP Layer	Protocol	Feature	Value
Network Interface	ARP	Destinations Per Source	[1,6]
		Requests Per Destinations	[1,365]
		Requests Per Source	[1,892]
		Responses Per Sources	[1,353]
		Responses Per Destination	[4,582]
Internet	ICMP	Destinations Per Source	[0,1]
		Requests Per Destinations	[0,4]
		Requests Per Source	[0,4]
		Responses Per Sources	[0,4]
		Responses Per Destination	[0,4]
	IGMP	Multicast Packets Per Second	[0,0.1]
		Unicast Packets Per Second	[0,0]
	–	LAND Indicator	0
Transport	TCP	TCP Packets	[284,18874]
		TCP Connections	[8,56]
		TCP Services	[1,5]
	UDP	UDP Packets	[132,534]
		UDP Connections	[31,144]
		UDP Services	[1,2]
Application	DNS	DNS NR	0
	NBNS	NBNS NR	0
	LLMNR	LLMNR NR	0

Table 7.10: Local Host-level Anomaly Features and Values

Table 7.10 is the same as Table 5.25 that we presented in Section 5.4.2, we just added the calculated feature values. These feature values represent the maximum boundaries of normal traffic for a given host, *i.e.*, thresholds. Thus, the traffic of anomalous hosts will violate one or more of these features.

These values are based on session-wise calculated numbers, as explained in Section 3.5. Thus, an important question arises here that might destroy the foundation of the proposed Local Host-level detection approach. How do we use

session-inferred values and compare them to minute-inferred ones? The answer to this question is simple, the concept of *Packet Capture Sessions*, explained in Section 6.2.1, is adopted to provide time-independent packet capture intervals. Therefore, the traffic of a session can be compared to the traffic of one minute, since they are both intervals, and both are not bound by a specific time unit. In addition, within our collected data we have *sessions* whose duration ranges from 15 seconds to 4 hours. To summarize, the process of comparing session-inferred thresholds to minute-calculated numbers is valid.

Once minute network traffic data record is detected as anomaly (by Global Network-level), we perform host-level investigation of minute traffic, Table 7.11.

Feature	Value	192.168.1.254	192.168.1.65	192.168.1.66	192.168.1.67	192.168.1.68	192.168.1.69
ARP Destinations Per Source	[1,6]	-	1	-	1	1	149
ARP Requests Per Destinations	[1,365]	-	-	-	1	1	3
ARP Requests Per Source	[1,892]	-	1	-	1	1	439
ARP Responses Per Sources	[1,353]	-	-	-	1	1	-
ARP Responses Per Destination	[4,582]	-	-	-	1	1	1
ICMP Destinations Per Source	[0,1]	-	-	-	-	-	2
ICMP Requests Per Destinations	[0,4]	-	-	-	-	-	51
ICMP Requests Per Source	[0,4]	-	-	-	-	-	52
ICMP Responses Per Sources	[0,4]	-	-	-	-	-	-
ICMP Responses Per Destination	[0,4]	-	-	-	-	-	-
IGMP Multicast Packets Per Second	[0,0.1]	0.03	0.03	-	0.05	0.016	0.03
IGMP Unicast Packets Per Second	[0,0]	-	-	-	-	-	-
LAND Indicator	0	-	-	-	-	-	-
TCP Packets	[284,18874]	30	36	-	34	-	-
TCP Connections	[8,56]	-	-	-	6	-	-
TCP Services	[1,5]	-	-	-	1	-	-
UDP Packets	[132,534]	-	1	-	18	-	-
UDP Connections	[31,144]	-	1	-	18	-	-
UDP Services	[1,2]	-	1	-	1	-	-
DNS NR	0	-	-	-	-	-	-
NBNS NR	0	-	-	-	-	-	-
LLMNR NR	0	-	-	-	-	-	-

Table 7.11: Host Traffic Example

Table 7.11 illustrates host-level investigation for the traffic of the minute example in Section 7.1.3. The table demonstrates Host-level features, their thresholds, the sources of traffic within the minute, and their related traffic. We can notice that the the aggregation of host traffic sums up to the minute traffic in Table 7.8.

Furthermore, it is clear that the traffic of host *'192.168.1.69'* is anomalous and violates normal thresholds of 1 ARP feature and 3 ICMP features. We end up

having Table 7.12 as result of Local Host-level Anomaly Detection.

IP Address	Violated Protocols	Violated Layers
192.168.1.69	ARP ICMP	Network Interface Internet

Table 7.12: Local Host-level Detection Result

One final note before we conclude this chapter, we can compare Table 7.12 to Table 3.13 and try to map this detected anomaly to an attack. We will find that the anomaly perfectly matches IPSweep attack, but this anomaly can also indicate ICMP Flood attack, since ICMP is violated (specifically, ICMP request packets). To conclude our Anomaly Detection context, we have detected an anomaly – Globally – and then we performed local Anomaly ID to gather more information. We now know which host is the source of the anomaly, what protocols were violated and which layers. We also know that the anomaly relates to ARP destinations, ICMP destinations, and ICMP request packets.

Chapter 8

Signature Intrusion Detection

There are multiple approaches used to identify network attacks. The most popular ones are matching traffic to attack signatures or constructing classifiers able to detect a single or a number of attacks. In the work presented here, the proposed Signature Detection system contains two steps:

Global Network-level Machine Learning-based Signature Detection Like in the case of Anomaly ID, the data representing a network traffic captured over a specified period of time – one minute in the presented work – is used as the input to a collection of classifiers to determine if an attack has taken place. If an attack is detected, the system goes to the next step, it analyzes attack signatures and provides detailed information about the attack. If an attack is not detected, the system indicates an anomalous state of the network. We utilize a number of Machine Learning techniques to construct classifiers: Decision Tree (DT), Logistic Regression (LR), and Support Vector Machine (SVM). We used a modified version of Logistic Regression called Simple Logistic (SL). Simple Logistic differs from Logistic Regression in having a built-in attribute selection technique, which enables it to produce better classification results than LR. Elements of Evidence Theory as applied to aggregate the obtained classifications.

Attack Detection Modules Unlike global network-level scope detection of attacks, attack detection modules utilize attack signatures to confirm and further analyze attacks, Section 8.2. We utilize updated network-tailored attack signatures to examine network traffic data from the points of view of packets and connection perspectives, and to match traffic patterns to attack signatures.

8.1 Global Network-level Machine Learning-based Signature Detection

The proposed Global Network-level Machine Learning-based Signature Detection approach is composed of three classifiers; DT, SL, and SVM, Figure 8.1. We select specific features from the anomalous network traffic data that relate to each attack according to the explanation presented in Sections 3.2 and 3.6. These features are passed to classifiers to process them and produce a classification result.

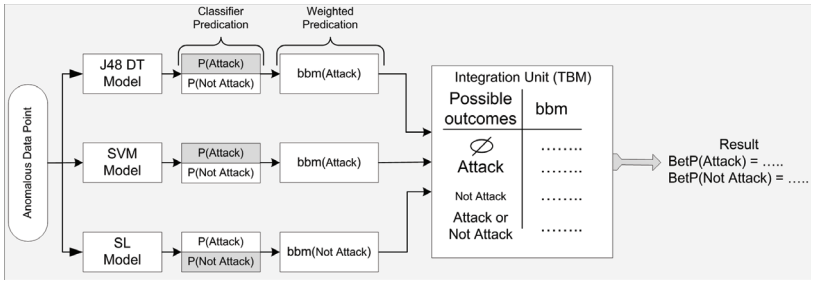


Figure 8.1: Architecture of Global Network-level Machine Learning-based Signature Detection Approach for Single Attack

Based on the prediction quality of each classifier, we assign belief masses to the classification result of each individual classifier. The weighted results of classifiers are passed to TBM as inputs, TBM in turn processes these inputs and

provides a probabilistic output that tells whether an attack exists or not. Such a process is performed for each attack from the selected set of attacks. As the result, we have seven TBMs, each corresponding to a single attack.

8.1.1 Implementation

Here we deal only with attacks, therefore we have extracted 322 attack data records from the training dataset. This data is used to construct classifiers. In essence, we do not create one classifier, for example DT, that distinguishes between all attack classes, but seven independent DT classifiers. Each such classifier assigns a data point to one of two classes: **AttackName** or **NoAttack**.
As the result, seven labels are assigned simultaneously for each minute data record. Each label corresponds to one of the attacks. Each classifier is trained for a specific attack based on 322 labeled data points. Collectively to train seven classifiers – one for each attack – we label each data point with seven labels. Table 8.1 depicts a few data points.

Minute Data Record	Anomaly ID	ICMP Flood Attack	IGMP Flood Attack	LAND Attack	Smurf Attack	IPSweep Attack	InsideSniffer Attack	PortScan Attack
1	Anomaly	NoAttack	IGMP Flood	LAND	NoAttack	NoAttack	NoAttack	NoAttack
2	Anomaly	NoAttack	NoAttack	NoAttack	NoAttack	IPSweep	NoAttack	NoAttack
3	Anomaly	NoAttack	NoAttack	NoAttack	NoAttack	IPSweep	InsideSniffer	NoAttack
4	Anomaly	ICMP Flood	NoAttack	NoAttack	NoAttack	IPSweep	NoAttack	NoAttack
5	Anomaly	ICMP Flood	NoAttack	NoAttack	Smurf	IPSweep	NoAttack	NoAttack

Table 8.1: Attack Label Example

The decision to have separate classifiers for each attack instead of one classifier for all attacks has been dictated by the ability to identify multiple attacks that can occur in the same time interval. Each data record, representing a network traffic over one minute (in our case) can represent several attacks, as illustrated in Table 8.1. In other words, each network traffic data record of one minute can hold multiple attack patterns simultaneously. For an individual

attack, we need to have a high quality classifer with the capability to distinguish a specific attack in the presence of other simultaneous attacks.

8.1.1.1 Implementation Details of Classifiers

In order to implement the Global Network-level Machine Learning-based Signature Detection as a module to our Intrusion Detection System, we have to embed different classifiers into our C#-based system. For the SVM, we have installed and utilized C# *libsvm* library. To create a model, *libsvm* requires two pieces of information: support vectors representing input data, and their corresponding labels. Once the library *libsvm* is available, it is easy to obtain needed information directly from the database and pass it to the library for processing.

For DT and SL, we export the attack training data from the database into CSV files. To construct these classifiers we use WEKA. Seven DTs have been produced, one for each attack. Similarly, seven logistic functions have been generated. We have integrated these DTs and functions with our system.

8.1.1.2 Construction of Classifiers

Once all classifiers are constructed, we determine their performance using both Training and Testing datasets. Based on the obtained classification results, we have calculated the performance measures to be used as belief masses associated with each classifier, Table 8.2. The subscript T represents the TRUE – original – type of Attack, while P means the PREDICTED type of Attack. Additionally, the symbol **A** represents **Attack**, while **NA** represents **NOT-Attack**.

	DT		**SL**		**SVM**	
	A_P	NA_P	A_P	NA_P	A_P	NA_P
A_T(368)	367	1	367	1	368	0
NA_T(1886)	1	1885	1	1885	0	1886
	AC:	99.9	AC:	99.9	AC:	100
	SP:	99.9	SP:	99.9	SP:	100
	SN:	99.7	SN:	99.7	SN:	100

(a) Performance of Classifiers with Training Dataset

	DT		**SL**		**SVM**	
	A_P	NA_P	A_P	NA_P	A_P	NA_P
A_T(84)	84	0	80	4	78	6
NA_T(364)	0	364	3	361	2	362
	AC:	100	AC:	98.4	AC:	98.2
	SP:	100	SP:	99.2	SP:	99.5
	SN:	100	SN:	95.2	SN:	92.9

(b) Performance of Classifiers with Testing Dataset

Table 8.2: Performance Measures of Signature Classifiers

It is clear from the above tables that the best performance for both datasets, is provided by DT, while SL comes next, and SVM provides a slightly lower performance than SL.

We have tested the following options for determining belief masses using the Evaluation Dataset:

Option A: using Bernoulli's Combination Rule;

Option B: taking only bbm^{TR};

Option C: taking only bbm^{TS};

Option D: averaging bbm^{TR}'s and bbm^{TS}'s.

We have found out that the TBM integration process gives the same results with all these options. Therefore, we adopt the Evidence Theory process and we use *Bernoulli's Combination Rule* to update belief masses.

8.1.2 Evaluation

We evaluate the performance of our proposed approach by analyzing the overall classification of the TBM (Section 8.1.2.1) and analysis of misclassified results (Section 8.1.2.2). The evaluation dataset contains 78 anomalous data records (Section 6.2.2). Each record is labeled with seven labels indicating occurrence or not of an individual attack. In total, we have 546 labels: 85 among them are attacks, and 461 are not attacks.

The detailed performance of DT, SL, and SVM against the Evaluation dataset and its included attacks is depicted in Tables 8.3, 8.4, and 8.5 respectively. As before, the subscript T represents the TRUE – original – type of Attack, while P means the PREDICTED type of Attack; the symbol **A** represents **Attack** class and **NA** represents **NOT-Attack**. We calculated the following performance measures: Accuracy (AC), Sensitivity (SN), Specificity, (SP), Precision (PR), and Recall (RC).

	DT	
	A_P	NA_P
A_T (85)	84	1
NA_T (461)	1	460
	SN:	98.8
	SP:	99.8
	AC:	99.6
	PR:	98.8
	RC:	98.8

(a) DT Overall Performance

	ICMP Flood			IGMP Flood			Smurf			LAND	
	A_P	NA_P		A_P	NA_P		A_P	NA_P		A_P	NA_P
A_T (20)	19	1	A_T (12)	12	0	A_T (8)	8	0	A_T (7)	7	0
NA_T (85)	0	85	NA_T (66)	0	66	NA_T (70)	0	70	NA_T (71)	0	71

(b) DT Performance Against Denial of Service Attacks

	IPSweep			InsideSniffer			PortScan	
	A_P	NA_P		A_P	NA_P		A_P	NA_P
A_T (26)	26	0	A_T (11)	11	0	A_T (1)	1	0
NA_T (52)	1	51	NA_T (67)	0	67	NA_T (77)	0	77

(c) DT Performance Against Probe Attacks

Table 8.3: Decision Tree Detailed Performance

Table 8.3a shows that DT misclassified two instances: one 'Attack' instance was misclassified as 'Not-Attack', and one 'Not-Attack' instance was misclassified as 'Attack'. Based on Tables 8.3b and 8.3c we determine that the misclassified 'Attack' instance is ICMP Flood attack, and the misclassified 'Not-Attack' instance has been classified as IPSweep attack. As we can see, the misclassifications are related to ICMP Flood and IPSweep attacks.

	SL	
	A_P	NA_P
A_T (85)	75	10
NA_T (461)	0	461
	SN:	88.2
	SP:	100
	AC:	98.2
	PR:	100
	RC:	88.2

(a) SL Overall Performance

	ICMP Flood			IGMP Flood			Smurf			LAND	
	A_P	NA_P		A_P	NA_P		A_P	NA_P		A_P	NA_P
A_T (20)	19	1	A_T (12)	12	0	A_T (8)	0	8	A_T (7)	7	0
NA_T (85)	0	85	NA_T (66)	0	66	NA_T (70)	0	70	NA_T (71)	0	71

(b) SL Performance Against Denial of Service Attacks

	IPSweep			InsideSniffer			PortScan	
	A_P	NA_P		A_P	NA_P		A_P	NA_P
A_T (26)	26	0	A_T (11)	11	0	A_T (1)	0	1
NA_T (52)	0	52	NA_T (67)	0	67	NA_T (77)	0	77

(c) SL Performance Against Probe Attacks

Table 8.4: Simple Logistic Detailed Performance

Likewise, Table 8.4 illustrates the detailed performance of SL when tested against the Evaluation dataset. Table 8.4a includes ten misclassified 'Attack' instances, which are classified as 'Not-Attack'. We can identify these misclassified attack instances by examining Table 8.4b and and Table 8.4c. As before, the misclassifications involve ICMP Flood, Smurf, and PortScan attacks. It is important to emphasize that SL could not properly identify any of the existing

Smurf attack instances.

	SVM	
	A_P	NA_P
A_T (85)	77	8
NA_T (461)	2	459
	SN:	90.6
	SP:	99.6
	AC:	98.2
	PR:	97.5
	RC:	90.6

(a) SVM Overall Performance

	ICMP Flood			IGMP Flood			Smurf			LAND	
	A_P	NA_P		A_P	NA_P		A_P	NA_P		A_P	NA_P
A_T (20)	20	0	A_T (12)	12	0	A_T (8)	0	8	A_T (7)	7	0
NA_T (85)	1	84	NA_T (66)	0	66	NA_T (70)	0	70	NA_T (71)	0	71

(b) SVM Performance Against Denial of Service Attacks

	IPSweep			InsideSniffer			PortScan	
	A_P	NA_P		A_P	NA_P		A_P	NA_P
A_T (26)	26	0	A_T (11)	11	0	A_T (1)	1	0
NA_T (52)	1	51	NA_T (67)	0	67	NA_T (77)	0	77

(c) SVM Performance Against Probe Attacks

Table 8.5: SVM Detailed Performance

Table 8.5 illustrates the detailed performance of SVM when tested against the Evaluation dataset. In Table 8.5a we can see ten misclassified instances: eight 'Attack' instances, and two 'Not-Attack' instances. From Tables 8.5b and 8.5c we can identify these misclassified instances; they involve ICMP Flood, Smurf, and IPSweep attacks.

The evaluation of classifiers brings the following conclusions:

- integration of multiple classifiers could provide better classification results, it is obvious from tables 8.3, 8.4, and 8.5;
- misclassified attacks are ICMP Flood, Smurf, and IPSweep that seems quite justified – the traffic patterns of all these attacks are based on ICMP (Section 3.2, Section 3.6).

It should be emphasized that we should achieve 100% attack detection accuracy.From the proposed system point of view, misclassified NOT-Attack instances do not constitute a significant problem. The next stage of the system – Attack Detection Modules – is a safe guard against such a situation. The Attack Detection Modules are able to detect existing attacks.

8.1.2.1 System Performance

Table 8.6 illustrates the performance of TBM against the Evaluation dataset.

	TBM	
	A_P	NA_P
A_T (85)	85	0
NA_T (461)	2	459
	SN:	100
	SP:	99.6
	AC:	99.6
	PR:	97.7
	RC:	100

(a) TBM Overall Performance

	ICMP Flood			IGMP Flood			Smurf			LAND	
	A_P	NA_P		A_P	NA_P		A_P	NA_P		A_P	NA_P
A_T (20)	20	0	A_T (12)	12	0	A_T (8)	8	0	A_T (7)	7	0
NA_T (85)	1	84	NA_T (66)	0	66	NA_T (70)	0	70	NA_T (71)	0	71

(b) TBM Performance Against Denial of Service Attacks

	IPSweep			InsideSniffer			PortScan	
	A_P	NA_P		A_P	NA_P		A_P	NA_P
A_T (26)	26	0	A_T (11)	11	0	A_T (1)	1	0
NA_T (52)	1	51	NA_T (67)	0	67	NA_T (77)	0	77

(c) TBM Performance Against Probe Attacks

Table 8.6: TBM Detailed Performance

The performance of TBM proves that we have reached our goal of achieving 100% detection accuracy for existing attacks, illustrated by *Sensitivity*. Various interesting observations can be done here. Looking just at the 'Attack' instances and comparing them to the results of individual classifiers, we can notice that

the TBM integration outperforms individual classifiers in the detection and classification of existing attacks.

On the other hand, when we investigate 'Not-Attack' instances, it seems that the misclassifications of TBM are similar to those of SVM, but this is not entirely true and will become obvious in the next section when we analyze the misclassified instances.

A final note here, the prevailing concept is that the final result of TBM is determined by comparing **BetP(A)** and **BetP(NA)** to the threshold of **0.5**. If **BetP(A)** exceeds 0.5 then the data record is '*Attack*', likewise, if **BetP(NA)** exceeds 0.5 then the given data record is '*Not-Attack*'. Now in order to obtain such performance, we change the threshold of 0.5 to the value of 0.65. This claim is justifiable for the following reasons:

1. All instances to be classified are anomalous in nature, thus we are trying to distinguish between even further anomalous patterns;
2. Multiple Attack instances are associated with the same data, it is different when compared to the conventional concept of having one class per data record;
3. Traffic of several attacks involves the same protocol(s);
4. An elusive execution pattern of one attack might resemble attack behaviors of other attacks.

8.1.2.2 Analysis of Misclassified Cases

The proposed approach has resulted in 2 misclassifications, Table 8.7.

Original Attack Classes	Attack Classes	DT	SL	SVM	**BetP(A)**	**BetP(NA)**
Smurf	ICMP Flood	0	0.063	1	1	0
	IGMP Flood	0	0	0	0.5	0.5
	Smurf	1	0.151	0	1	0
	LAND	0	0.02	0	0.51	0.49
	IPSweep	0	0.067	0	0.53	0.47
	InsideSniffer	0	0.223	0	0.61	0.39
	PortScan	0	0.061	0	0.53	0.47
ICMP Flood	ICMP Flood	1	1	1	1	0
	IGMP Flood	0	0	0	0.5	0.5
	Smurf	0	0	0	0.5	0.5
	LAND	0	0.02	0	0.51	0.49
	IPSweep	1	0.404	1	1	0
	InsideSniffer	0	0.223	0	0.61	0.39
	PortScan	0	0.061	0	0.53	0.47

Table 8.7: Misclassified Data Points with TBM Signature ID

The first data record originally contained Smurf attack, TBM classified it correctly and it also concluded that it contains ICMP Flood. We can notice that the governing classifiers for these results are DT and SVM respectively. DT classified it as Smurf and SVM did not, taking into account the non-zero (0.151) output of SL, shifts the belief of TBM to the correct classification Smurf. In the same manner, the non-zero (0.063) result of SL, besides the misclassification of SVM, biased TBM to misclassify the point as ICMP Flood.

For the second misclassified data record, all three classifiers correctly and strongly concluded that it contains ICMP Flood. However, DT and SVM backed up by the 0.404 probability of SL rendered the final misclassification, i.e., the data record has been also classified as having the IPSweep attack.

8.1.3 Illustrative Example

Referring to the example provided in Table 7.8, the minute values are passed to the classifiers and TBM for classification. We have obtained the result depicted in Table 8.8

Original Attack Classes	Attack Classes	DT	SL	SVM	**BetP(A)**	**BetP(NA)**
ICMP Flood IPSweep	ICMP Flood	1	0.838	1	1	0
	IGMP Flood	0	0	0	0.5	0.5
	Smurf	1	0.068	0	0.53	0.47
	LAND	0	0.02	0	0.51	0.49
	IPSweep	1	1	1	1	0
	InsideSniffer	0	0.223	0	0.61	0.39
	PortScan	0	0.045	0	0.52	0.48

Table 8.8: SVM Classification for give Example

According to our proposed approach, the minute traffic contains ICMP Flood attack and IPSweep attack, and yes, in this minute we simulated these attacks. The proposed approach was able to correctly classify the existing attacks. Now, these results confirm the initial findings that we concluded earlier in Local Host-level Anomaly ID, Section 7.2, regarding the anomalously violated layers and protocols.

8.2 Attack Detection Modules

In this section, we explain how the updated network-tailored attack signatures (explained in Section 3.6) are used to detect attacks. The matching process of network traffic to attack signatures is done by examining packets and connections.

For all attacks to be explained, after extracting attack traffic of a certain cyber attack, we start to follow the attack equation explained in Section 3.2.8.1 to determine the severity of the detected attacks. If one attack instance is detected, it is considered as basic attack severity. Otherwise, if more than one attack instance are detected, then the attack is sever and and we consider it as elusive attack.

Before we continue with our explanation of attacks, we need to mention something important. In our selected set of attacks we have ICMP Flood, Smurf,

and IPSweep attack. ICMP Flood attack and Smurf attack are Denial of Service attacks that are executed to starve the target machine. So, to successfully execute these attacks we need to have high volumes of packets and connections, which has to be issued in a small duration of time. On the Other hand, IPSweep attack is a Probe attack that performs scanning and reconnaissance activity. The success of this attack does not depend on high volumes of packets and connections, and in the same time can take longer durations.

We mention these attacks specifically because their execution pattern might cause confusion to implemented IDSs, since all of them rely on the same protocol: ICMP. So a distinction has to be made between the traffic patterns of these attacks. Mainly with DOS attacks, we are looking for high volume traffic (packets and/or connections) with ICMP. ICMP Flood is characterized by ICMP ping request packets, while Smurf is identified by ICMP ping reply packets. Contrarily, IPSweep indicates low or medium volumes of ICMP, using ICMP ping requests or both requests and responses.

8.2.1 ICMP Flood Attack Detection Module

Referring back to Sections 3.2.1 and 3.6.1, it is clear that the flooding effect of the attack can be executed using a high number of ICMP ping request packets. These packets might or might not result in a correspondingly high number of ICMP connections. These connections 'must' have a status (Section 5.3.1) of either *'Complete'* or *'Incomplete E'*. All of this has to be present within the traffic in relatively short time interval. Thus, The first step of ICMP Flood attack detection module is to search within network traffic and find those patterns which fit these constraints.

Following the attack equation, we start by investigating the sources, destinations,

packets and connections within the extracted attack traffic. One attack source and one attack target signifies an attack instance, which is in turn called basic attack. Otherwise, ICMP Flood is called elusive attack and more processing is made to detect the number of attack instances between attack sources and their respective targets.

To illustrate the idea for the reader, we give a simple example. Within our captured data we have an overlay session (session 1850) which contains ICMP Flood attack traffic that is superimposed on normal traffic. Our ICMP Flood attack detection module was able to detect ICMP Flood attack within that session, Figure 8.2

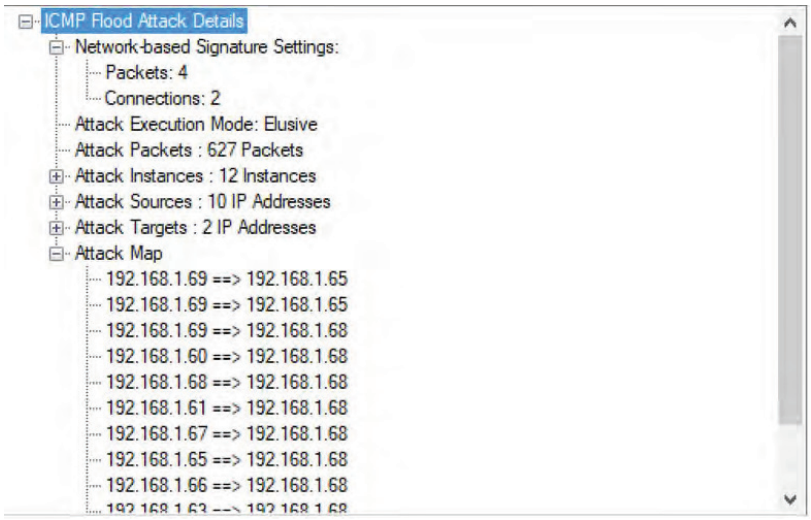


Figure 8.2: ICMP Flood Attack Detection Module Summary Detection Information

As illustrated, we use updated network-tailored ICMP Flood signature (Section 3.6.1). The module was able to detect 627 attack packets belonging to 12 attack instances. Also, it was able to detect that the attack was executed by a total of 10 attackers against 2 targets. The figure also represents a simple attack map that designates which attacker attacked which target. Let's examine the first

2 attack instances for example, they have the same attacker and the same target, Figure 8.3.

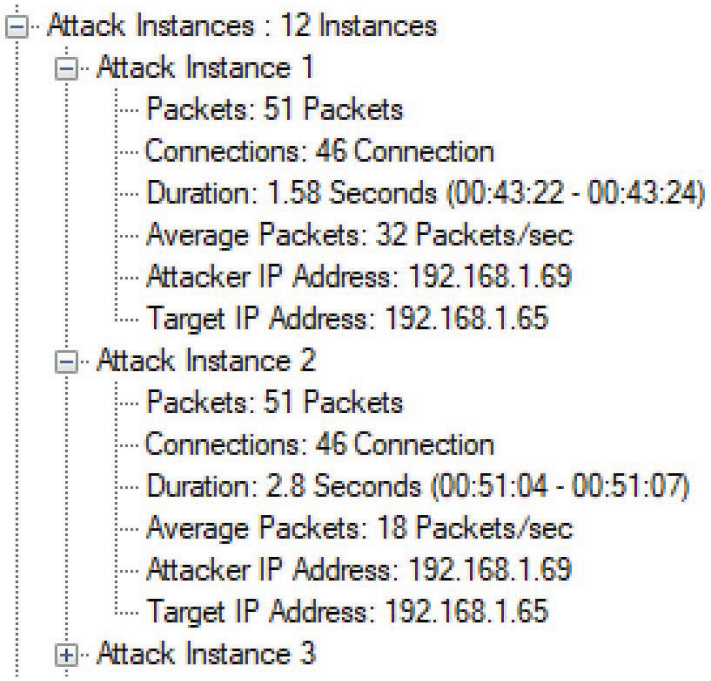


Figure 8.3: ICMP Flood Attack Instances 1 and 2

How did our detection module detect that the traffic from *'192.168.1.69'* to *'192.168.1.65'* constitutes 2 attack instances not 1 instance? The answer is simple, we examine the packet Timestamps. If we look closely at Figure 8.3, namely the duration criteria of each instance, we will find that there is a time gap of nearly 7.6 minutes (exactly 460 seconds) between the 2 instances, where instance 1 took 1.58 seconds from 00:43:22 to 00:43:24 and instance 2 took 2.8 seconds from 00:51:04 to 00:51:07.

Also, comparing these value to the updated network-tailored ICMP Flood signature (Section 3.6.1), we can easily see, by a simple comparison, that for signature feature *'3.Transmitted ICMP request packets per distinct hosts'*, a number of transmitted 51 packets in attack instances fulfills the signature feature

compared to 4 packets. Also, signature feature *'4.Established ICMP connections per distinct hosts'* the number 46 connections for attack instances satisfies the feature compared to two connections.

8.2.2 IGMP Flood Attack Detection Module

Referring back to sections 3.2.2 and 3.6.2, the flooding effect of the attack can be executed using a high number of IGMP unicast packets. These packets has to be executed within a relatively short time interval. IGMP Flood attack detection module searches within network traffic to find IGMP unicast traffic patterns.
Once IGMP Flood attack traffic is extracted we follow the attack equation and start investigating the sources, destinations, and packets to find attack instances. Within the same session, session 1850, our proposed IGMP Flood attack detection module was able to detect IGMP Flood attack, Figure 8.4.

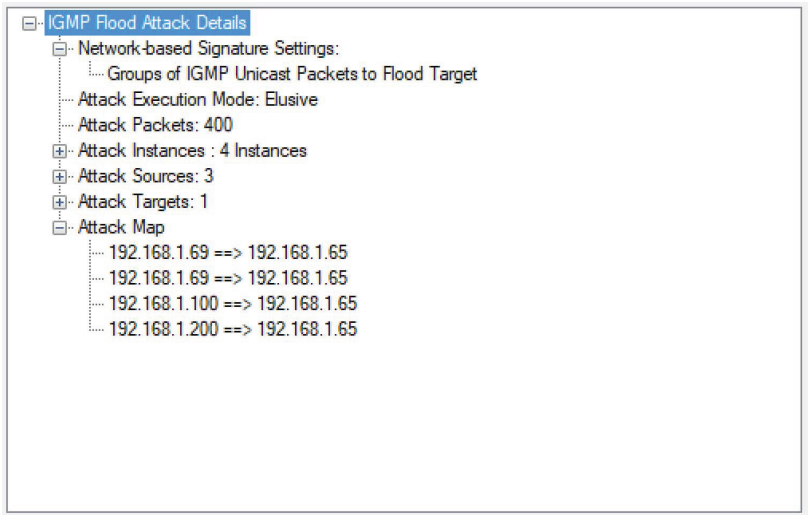


Figure 8.4: IGMP Flood Attack Detection Module Summary Detection Information

Again as presented, we use updated network-tailored IGMP Flood signature (Section 3.6.2). The module was able to detect a total of 400 attack packets

belonging to 4 attack instances, executed by a total of 3 attackers against 1 target, the attack map designates such attackers and target. Surprisingly, the first 2 attack instances are similar to those of ICMP Flood having the same attacker (192.168.1.69) and the same target (192.168.1.65), Figure 8.5.

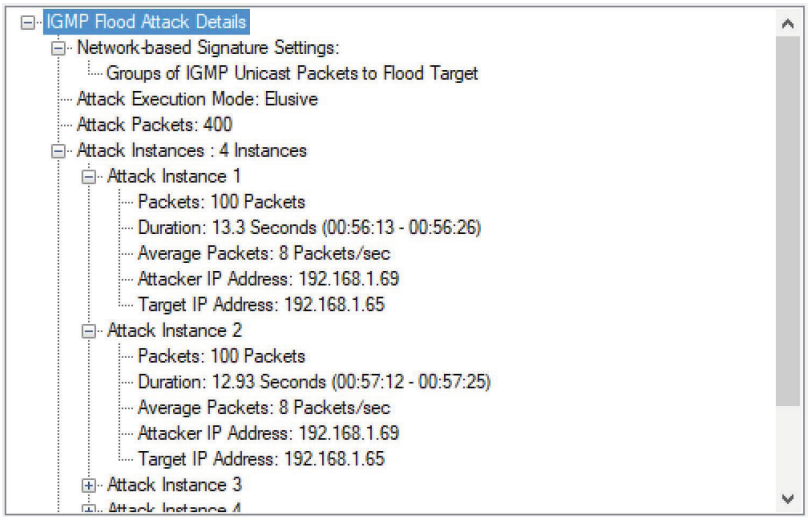


Figure 8.5: IGMP Flood Attack Instances 1 and 2

Instance 1 was executed using 100 packets and elapsed a duration of 13.3 seconds while instance 2 was executed using 100 packets over a time duration of 12.93 seconds. Though, instances 1 and 2 have the same attacker and the same target, one can easily conclude that they are 2 instances not 1 instance by examining the duration criteria of each (start and end timestamps).

Comparing attack instances and their values to those of updated network-tailored IGMP Flood signature (Section 3.6.2), we can easily recognize IGMP Flood attack, where signature feature *'Transmitted IGMP unicast packets per distinct host'* is satisfied by the instances having 100 IGMP unicast packets instead of zero.

8.2.3 Smurf

Sections 3.2.3 and 3.6.3 state that the flooding effect of Smurf attack is recognized by having 32 ICMP ping response packets or 16 ICMP connections whose status is *Incomplete B* (connectionIPv4), present within network traffic in a small time duration. Smurf attack detection module searches for such patterns and extracts them.

Once Smurf attack traffic is extracted we apply the attack equation and start investigating the sources, destinations, packets, and connections to find attack instances. Within session 1860, our proposed Smurf attack detection module was able to detect Smurf attack, Figure 8.6.

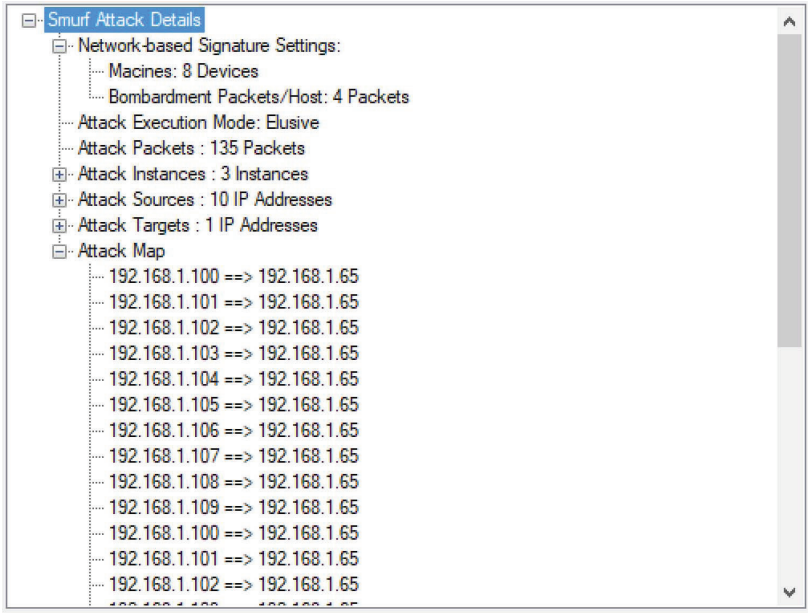


Figure 8.6: Smurf Attack Detection Module Summary Detection Information

Using updated network-tailored Smurf signature (Section 3.6.3), the module was able to detect a total of 135 attack packets belonging to 3 attack instances, executed by 10 attackers against 1 target, the attack map designates that the range of Addresses *192.168.1.100 - 192.168.1.109* are the 10 attackers and the target is

192.168.1.65. It is somehow clear according to Smurf attack definition that these addresses flooded the same target 3 consecutive times. Question, how do we see 10 attackers and 1 target when our network includes only 8 machines? Easy, the attacker used botnets to attack the target. Figure 8.7 illustrates the details of the first 2 attack instances.

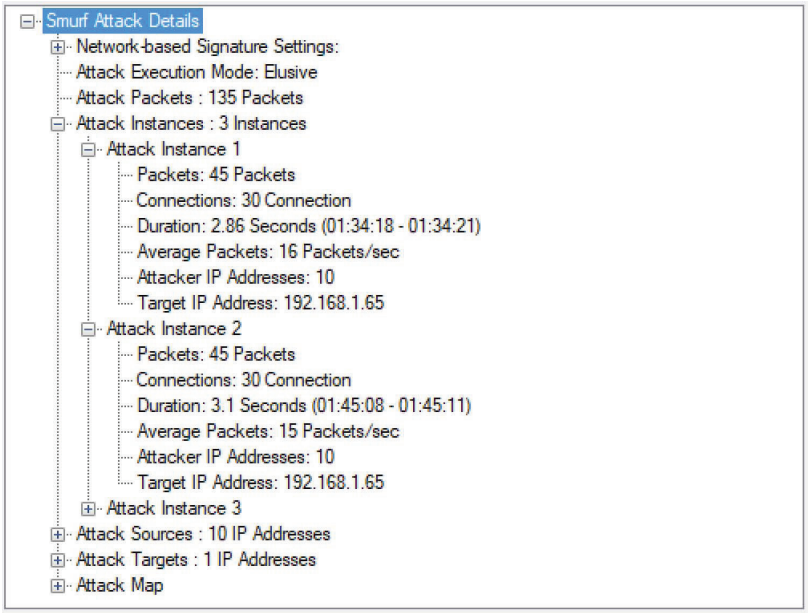


Figure 8.7: IGMP Flood Attack Instances 1 and 2

Instance 1 was executed using 45 packets and 31 connections in a duration of 2.86 seconds, and instance 2 using the exact number of packets and connections elapsed for a duration of 3.1 seconds. Examining packet timestamps, we calculate the duration of each instance and we distinguish the instances by detecting a time gap of 10.8 minutes (647 seconds).

Comparing attack instances and their values to those of updated network-tailored Smurf signature (Section 3.6.3), we can easily notice Smurf attack, where signature feature *'5.Overall Number of attack packets'* has been met for both instances, 45 packets for attack instances compared to 32 packets.

Also, signature feature *'6.Overall Number of attack connections'* has been accomplished for both instances; 30 connections for attack instances compared to 16 connections.

One important note to be mentioned here, from the attack instances we can concur that each attacker transmitted 15 packets, constituting 10 connections. These attack execution settings are low for Smurf attack, and we intentionally executed them in this manner to try to mimic an elusive attack which slightly deviates from normal. We wanted to prove that the proposed system has the ability to detect the most elusive attacks.

8.2.4 LAND

According to the LAND attack analysis provided in sections 3.2.4 and 3.6.4, any recognized LAND attack traffic is extracted and the attack equation applied to find attack instances. Within session 1850, our proposed LAND attack detection module was able to detect LAND attack, Figure 8.8.

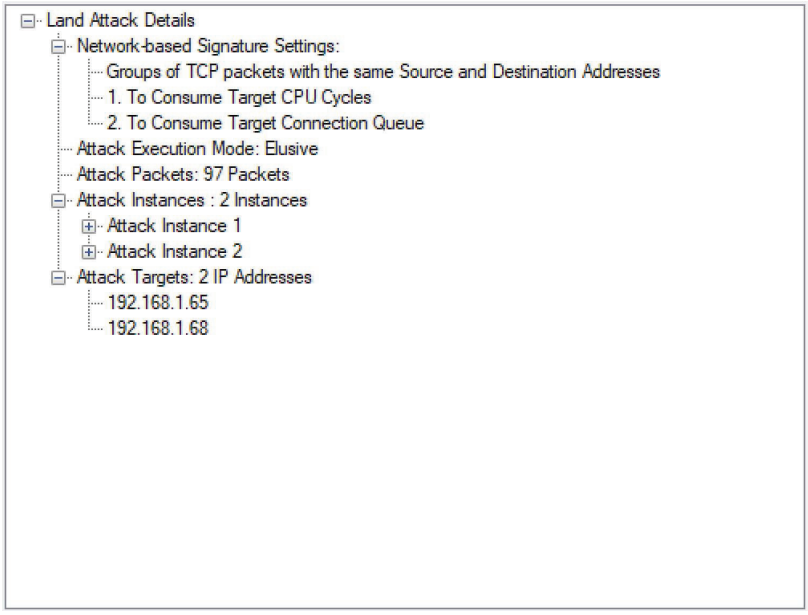


Figure 8.8: LAND Attack Detection Module Summary Detection Information

Using updated network-tailored LAND signature, the module was able to detect a total of 97 attack packets belonging to 2 attack instances, executed against 2 targets, the attack map shows that the targets are *192.168.1.68* and *192.168.1.65*. Figure 8.9 illustrates the details of attack instances.

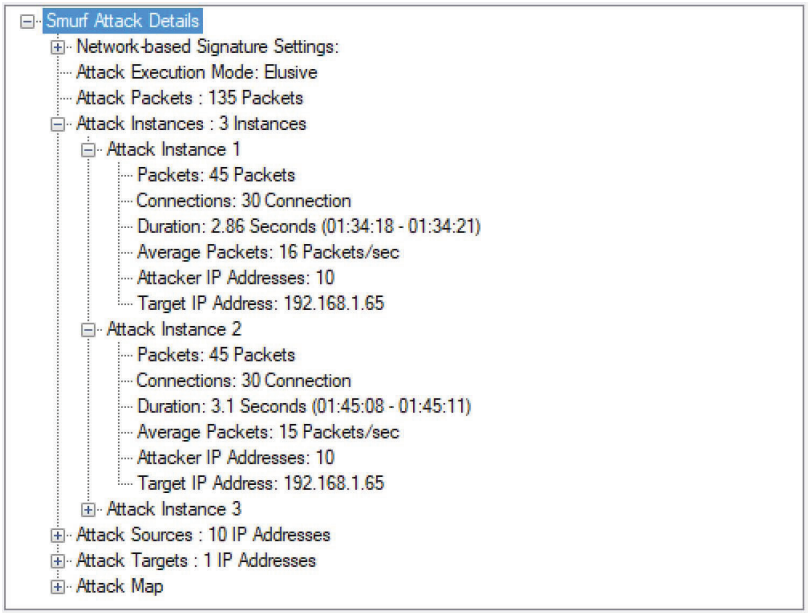


Figure 8.9: LAND Attack Instances 1 and 2

Instance 1 was executed using 48 packets and 1 connection in a duration of 1.67 seconds, and instance was executed using 49 packets and 1 connection and connections elapsed for a duration of 16.12 seconds. We can also notice that the protocol in the 2 instances is different, instance 1 used TCP protocol and instance 2 used ICMP protocol. It is also noticeable from the duration that instance 1 was targeting CPU cycles of the target and instance 2 was targeting the connection queue.

Comparing attack instances and their values to those of updated network-tailored LAND signature, we can easily conclude that LAND attack instances satisfy the signature.

One important note to be mentioned here, observing the details of attack instances we will find that the MAC address is Botnet. In the module we implemented a nice functionality that can match logical address to its physical, if a match is not found then this indicates a botnet.

8.2.5 IPSweep

Sections 3.2.5 and 3.6.5 state that the scan effect of IPSweep attack is recognized by an attacker scanning up to 7 addresses. IPSweep attack detection module searches for such patterns and extracts them.

Once IPSweep attack traffic is extracted we apply the attack equation and start investigating the sources, destinations, packets, and connections to find attack instances. Within session 1880, our proposed IPSweep attack detection module was able to detect IPSweep attack, Figure 8.10.

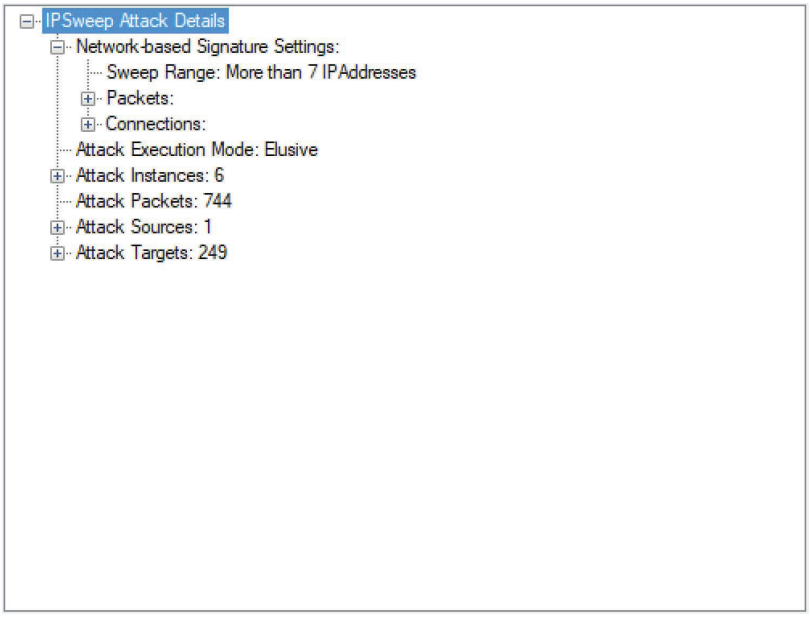
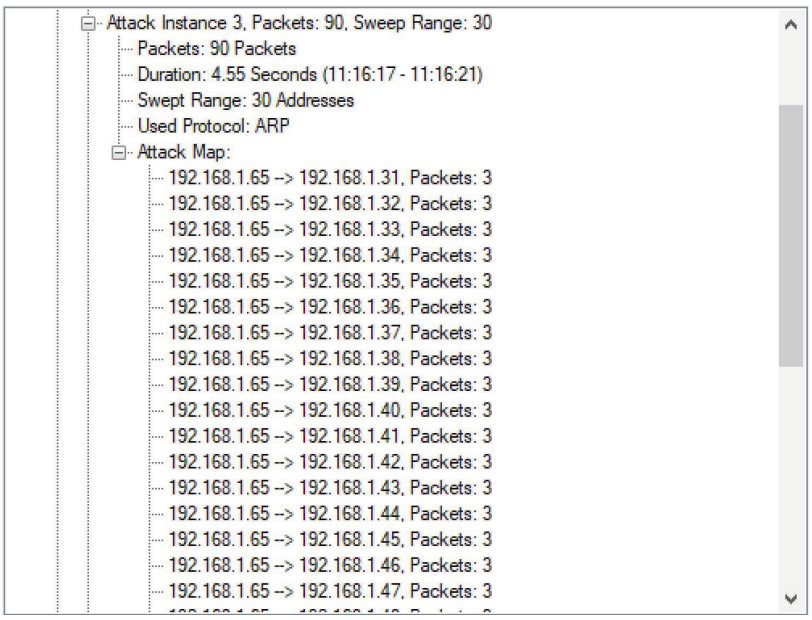
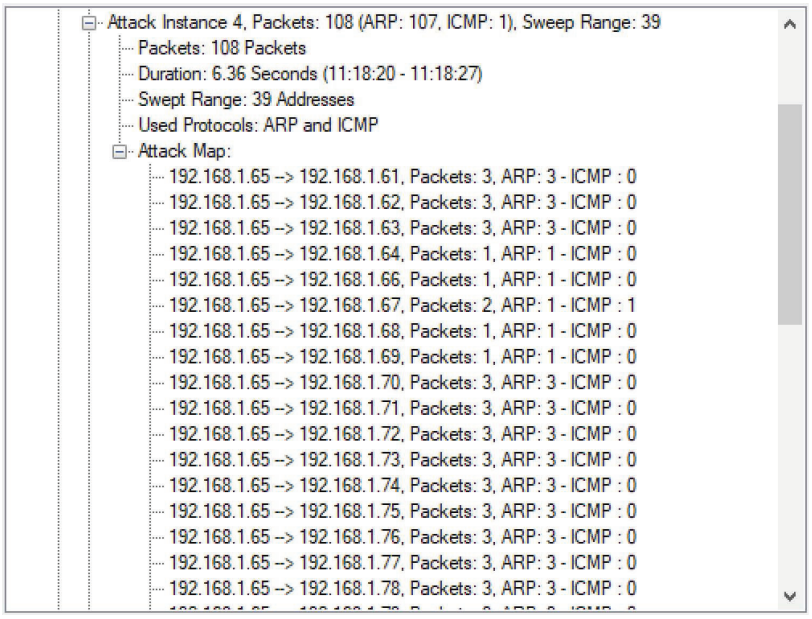


Figure 8.10: IPSweep Attack Detection Module Summary Detection Information

Using updated network-tailored IPSweep signature, the module was able to detect a total of 744 attack packets (743 ARP and 1 ICMP) belonging to 6 attack instances, executed by 1 attacker to scan 249 targets. Figure 8.11 shows sample IPSweep attack instances.



(a) IPSweep Attack Instance 3.



(b) IPSweep Attack Instance 4.

Figure 8.11: Sample IPSweep Attack Instances

Instance 3 (Figure 8.11a) was executed using 90 ARP packets in a duration of 4.55 seconds to scan 30 addresses (*192.168.1.31 - 192.168.1.60*), and instance 4 (Figure 8.11b) was executed using 108 packets; 107 ARP packets and 1 ICMP

packet, which elapsed for a duration of 6.36 seconds to scan 39 addresses (*192.168.1.61 - 192.168.1.100*). Examining packet timestamps, we calculate the duration of each instance and we distinguish between instances by detecting a time gap of nearly 2 minutes (119 seconds).

Comparing attack instances and their values to those of updated network-tailored IPSweep signature, we conclude that IPSweep attack instances satisfy the signature.

8.2.6 InsideSniffer

According to the InsideSniffer attack analysis provided in sections 3.2.6 and 3.6.6, InsideSniffer attack traffic includes DNS, NBNS, LLMNR protocols where a resolution from address to name occurs. The attack traffic is extracted and the attack equation is applied to find attack instances. Within session 1880, our proposed InsideSniffer attack detection module was able to detect 1 attack instance, Figure 8.12.

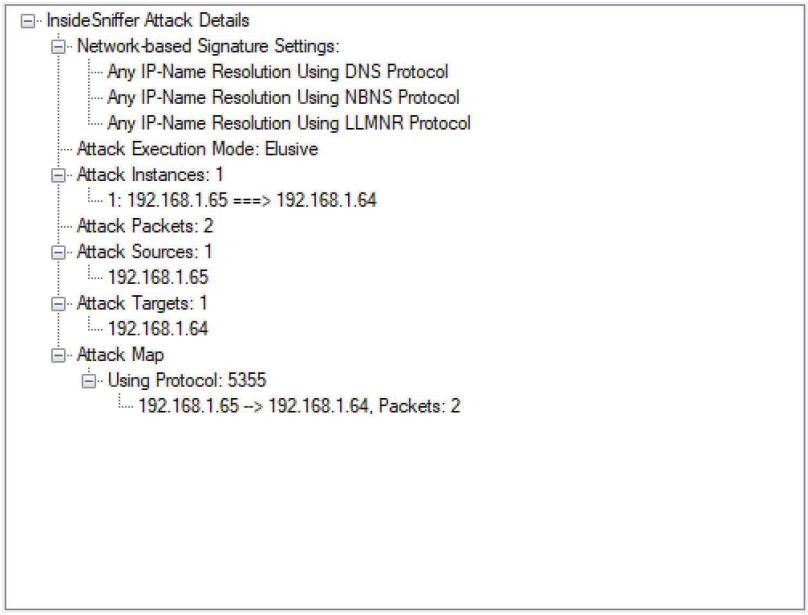


Figure 8.12: InsideSniffer Attack Detection Module Summary Detection Information

Using updated network-tailored InsideSniffer signature, the module was able to detect 2 attack packets from *192.168.1.65* using LLMNR protocol to scan 192.168.1.64 to its respective name. The detected attack instance conforms with updated network-tailored InsideSniffer signature.

8.2.7 Portscan

According to the PortScan attack analysis provided in sections 3.2.7 and 3.6.7, PortScan attack traffic is distinguished by the number of invoked services between 2 host over TCP. The attack traffic is identified and extracted and the attack equation is applied to find attack instances. Within session 120, our proposed InsideSniffer attack detection module was able to detect 1 attack instance, Figure 8.13.

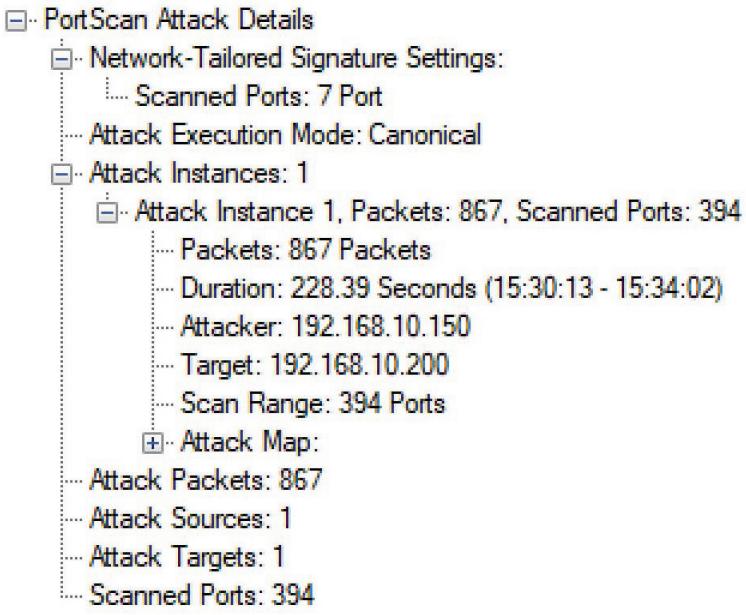


Figure 8.13: PortScan Attack Detection Module Summary Detection Information

Using updated network-tailored PortScan signature, the module was able to detect 1 attack instance from *'192.168.10.150'* against *'192.168.10.200'* to scan its ports. The attacker scanned 394 ports with 867 packets in a duration of 3.8 minutes (228.39 seconds).

Chapter 9

Hybrid Intrusion Detection: Case Studies

In this chapter, we provide a detailed examination of several Intrusion Detection scenarios. We choose a certain minute in our traffic then expose it to our system, module by module, and demonstrate the detection results. We demonstrate the performance of the system for three different minutes: the first minute is normal (Section 9.1); the second minute is an anomalous minute that contains LAND and IPSweep attack (Section 9.2); the third minute (Section 9.3) is anomalous and it is the most complex minute that we have within our traffic, it contains 4 simultaneous attacks.
Additionally, we provide an evaluation of our methodology – two stage detection of intrusion – using widely available NSL-KDD data, (Section 9.4).

9.1 Intrusion Detection Scenario 1: Normal Minute

Table 9.1 illustrates the traffic features of the normal minute.

TCP/IP Layer	Protocol	Feature	Minute
Network Interface	ARP	ARPRequests	3
		ARPResponses	0
		ARPSourceIP	2
		ARPDestinationIP	3
Internet	ICMP	ICMPQueryReq	0
		ICMPQueryRes	0
		ICMPSourceIP	0
		ICMPDestinationIP	0
		ICMPConnections	0
	IGMP	IGMPPackets	5
		IGMPMulticastP	5
		IGMPUnicastP	0
	–	LAND	0
Transport	TCP	TCPPackets	112
		TCPConnections	9
		TCPServices	4
	UDP	UDPPackets	84
		UDPConnections	18
		UDPServices	3
Application	DNS	DNSRequests	0
		DNSResponses	0
		DNSConnections	0
		DNSMulticastD	0
		DNSUnicastD	0
		DNSNR	0
	NBNS	NBNSRequests	1
		NBNSResponses	0
		NBNSConnections	1
		NBNSMulticastD	1
		NBNSUnicastD	0
		NBNSNR	0
	LLMNR	LLMNRRequests	4
		LLMNRResponses	0
		LLMNRConnections	4
		LLMNRMulticastD	4
		LLMNRUnicastD	0
		LLMNRNR	0

Table 9.1: Normal Minute Traffic (NetDataCoP).

We pass the minute to our system's first stage, *i.e.*, *Global Network-level Machine Learning-based Anomaly Detection.*

		K-Means	DT	SVM	LR	MAX	**BetP(N)**	**BetP(A)**
org-N	pred-N	0.5	0.87	0.75	0.8	0.92	0.9998	0.0001
	pred-A	–	–	–	–	–		

Table 9.2: Global Anomaly Detection: Normal Minute (NetDataCoP).

According to the results in Table 9.2, the minute was classified as normal. All the classifiers predicted the minute as normal and also TBM's final result of 0.9998 predicts it as normal.

9.2 Intrusion Detection Scenario 2: Attack Minute 1

Table 9.3 illustrates the traffic features of Attack Minute 1.

TCP/IP Layer	Protocol	Feature	Minute
Network Interface	ARP	ARPRequests	461
		ARPResponses	2
		ARPSourceIP	3
		ARPDestinationIP	53
Internet	ICMP	ICMPQueryReq	0
		ICMPQueryRes	0
		ICMPSourceIP	0
		ICMPDestinationIP	0
		ICMPConnections	0
	IGMP	IGMPPackets	10
		IGMPMulticastP	10
		IGMPUnicastP	0
	–	LAND	1
Transport	TCP	TCPPackets	212
		TCPConnections	15
		TCPServices	5
	UDP	UDPPackets	100
		UDPConnections	20
		UDPServices	2
Application	DNS	DNSRequests	2
		DNSResponses	2
		DNSConnections	2
		DNSMulticastD	0
		DNSUnicastD	4
		DNSNR	0
	NBNS	NBNSRequests	0
		NBNSResponses	0
		NBNSConnections	0
		NBNSMulticastD	0
		NBNSUnicastD	0
		NBNSNR	0
	LLMNR	LLMNRRequests	0
		LLMNRResponses	0
		LLMNRConnections	0
		LLMNRMulticastD	0
		LLMNRUnicastD	0
		LLMNRNR	0

Table 9.3: Attack Minute 1 Traffic (NetDataCoP).

Again, we pass the minute to Anomaly Detection to find out if the minute is normal or anomalous.

		K-Means	DT	SVM	LR	MAX	**BetP(N)**	**BetP(A)**
org-N	pred-N	–	–	–	–	–	0.0001	0.9998
	pred-A	*0.5*	*0.87*	*0.75*	*0.8*	*0.92*		

Table 9.4: Global Anomaly Detection: Attack Minute 1 (NetDataCoP).

Within *Global Network-level Machine Learning-based Anomaly Detection*, the classifiers and TBM (Table 9.4) classified the minute as anomalous, so we infer

additional information about the detected anomaly by investigating the minute traffic from *Local Host-level* scope.

IP Address	Protocol	Layer
192.168.1.69	ARP	Network Interface
192.168.1.65	IPv4	Internet

Table 9.5: Local Anomaly Detection: Attack Minute 1 (NetDataCoP).

Our *Local Host-level Threshold-based Anomaly Detection*, Table 9.5, detected that the traffic of the anomalous minute contains anomalies that relate to *Network Interface* Layer and *Internet* Layer. The anomaly within the Network Interface layer involved *ARP*, and that of Internet Layer did not involve a specific protocol.

Afterwards, the anomalous minute is passed to the second stage of the proposed system; Signature Detection.

Original Attack Classes	Attack Classes	DT	SL	SVM	**BetP(A)**	**BetP(NA)**
ICMP Flood IPSweep	ICMP Flood	0	0.072	0	0.54	0.46
	IGMP Flood	0	0	0	0.5	0.5
	Smurf	0	0.053	0	0.53	0.47
	LAND	1	1	1	1	0
	IPSweep	1	1	1	1	0
	InsideSniffer	0	0.223	0	0.61	0.39
	PortScan	0	0.061	0	0.53	0.47

Table 9.6: Global Signature Detection: Attack Minute 1 (NetDataCoP).

When we subjected the anomalous minute to our *Global Network-level Machine Learning-based Signature Detection*, it was able to detect LAND and IPSweep attacks, as indicated in Table 9.6 by the **BetP(A)** values of these attacks. We signal the Detection Modules of the detected attacks so that they detect their attack signatures within the minute traffic, Figure 9.1.

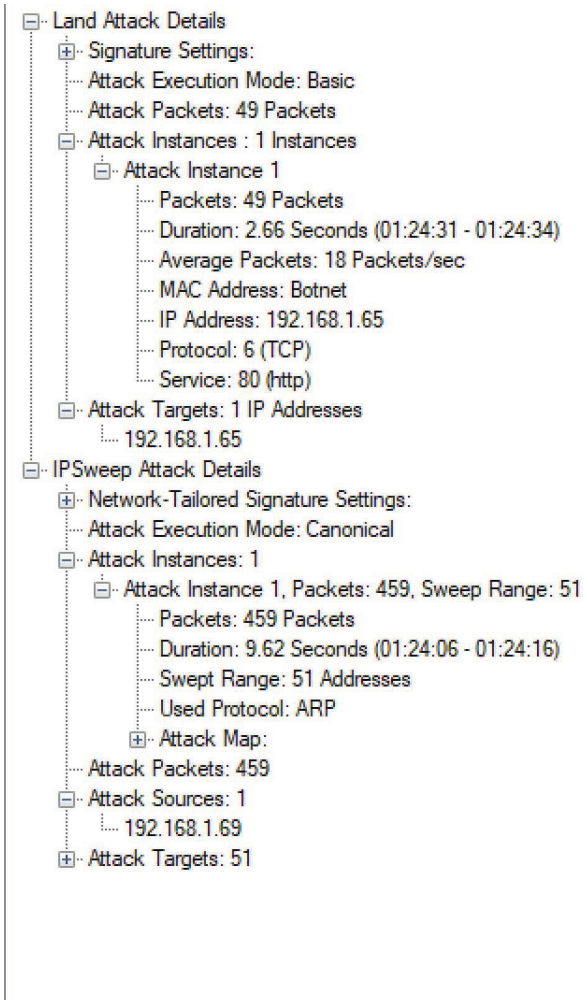


Figure 9.1: Attack Detection Modules: Attack Minute 1 (NetDataCoP).

The Attack Detection Modules processed the minute traffic and got the results depicted in Figure 9.1. From the figure, the LAND Detection Module was able to detect 1 attack instance, involving 49 packets, in a duration of 2.66 seconds within the minute, and its target is *'192.168.1.65'*. Also, the IPSweep Detection Module was able to detect 1 attack instance, executed using 459 ARP packets, in a duration of 9.62 seconds, by the attacker *'192.168.1.69'* to scan a range of 51 IP addresses. These results accurately conform with those obtained in Local Host-level Anomaly Detection, Table 9.5.

9.3 Intrusion Detection Scenario 3: Attack Minute 2

The traffic of this minute is shown in Table 9.7.

TCP/IP Layer	Protocol	Feature	Minute
Network Interface	ARP	ARPRequests	0
		ARPResponses	0
		ARPSourceIP	0
		ARPDestinationIP	0
Internet	ICMP	ICMPQueryReq	50
		ICMPQueryRes	45
		ICMPSourceIP	11
		ICMPDestinationIP	2
		ICMPConnections	75
	IGMP	IGMPPackets	305
		IGMPMulticastP	5
		IGMPUnicastP	300
	–	LAND	1
Transport	TCP	TCPPackets	165
		TCPConnections	9
		TCPServices	6
	UDP	UDPPackets	78
		UDPConnections	13
		UDPServices	1
Application	DNS	DNSRequests	0
		DNSResponses	0
		DNSConnections	0
		DNSMulticastD	0
		DNSUnicastD	0
		DNSNR	0
	NBNS	NBNSRequests	0
		NBNSResponses	0
		NBNSConnections	0
		NBNSMulticastD	0
		NBNSUnicastD	0
		NBNSNR	0
	LLMNR	LLMNRRequests	0
		LLMNRResponses	0
		LLMNRConnections	0
		LLMNRMulticastD	0
		LLMNRUnicastD	0
		LLMNRNR	0

Table 9.7: Attack Minute 2 Traffic

The minute features are passed to our first detection stage; *Anomaly Detection*, to decide whether the minute is normal or not.

		K-Means	DT	SVM	LR	MAX	**BetP(N)**	**BetP(A)**
org-N	pred-N	–	–	–	–	–	0.0001	0.9998
	pred-A	*0.5*	*0.87*	*0.75*	*0.8*	*0.92*		

Table 9.8: Global Anomaly Detection: Attack Minute 2 (NetDataCoP).

Our implemented *Global Network-level Machine Learning-based Anomaly Detection*, (Table 9.8), was able to classify the minute as anomalous, our next

step is to infer more information about this anomalous minute.

Anomalous Hosts	Violated Protocols	Violated Layers
192.168.1.65	ICMP IPv4	Internet
192.168.1.69	ICMP IGMP	
192.168.1.100	IGMP	
192.168.1.200	IGMP	

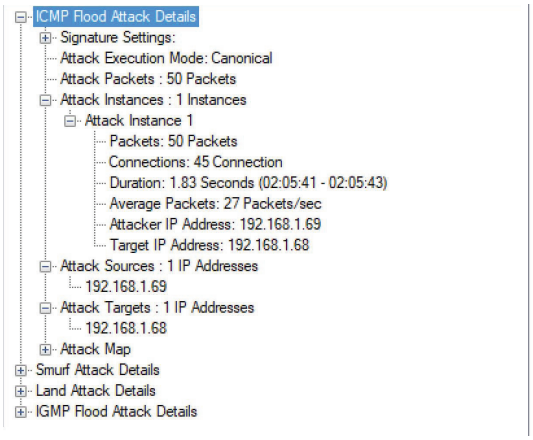
Table 9.9: Local Anomaly Detection: Attack Minute 2 (NetDataCoP).

The *Local Host-level Threshold-based Anomaly Detection*, Table 9.9, detected that the traffic of the anomalous minute contains anomalies that relate to *Internet* Layer. The detected anomaly involves ICMP, IGMP, and IPv4. Also, it detected that there are 4 anomalous sources responsible for the anomaly. After that, we pass the anomalous minute to the second stage of the proposed system; *Signature Detection*.

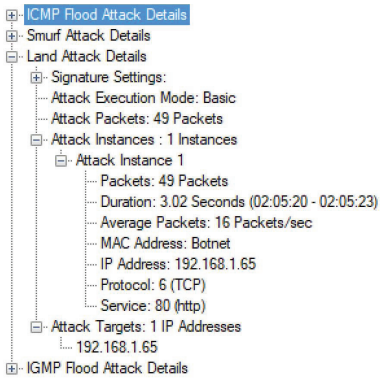
Original Attack Classes	Attack Classes	DT	SL	SVM	**BetP(A)**	**BetP(NA)**
ICMP Flood **IGMP Flood** **Smurf** **LAND**	ICMP Flood	1	0.853	1	1	0
	IGMP Flood	1	1	1	1	0
	Smurf	1	0.48	0	1	0
	LAND	1	1	1	1	0
	IPSweep	0	0.067	0	0.53	0.47
	InsideSniffer	0	0.223	0	0.61	0.39
	PortScan	0	0.071	0	0.54	0.46

Table 9.10: Global Signature Detection: Attack Minute 2 (NetDataCoP).

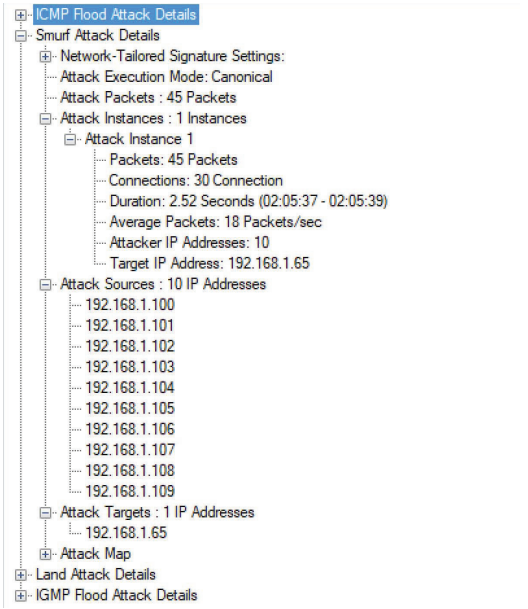
The *Global Network-level Machine Learning-based Signature Detection*, Table 9.10, was able to detect 4 concurrent attacks; ICMP Flood, Smurf, LAND and IGMP Flood attack, as indicated by their respective **BetP(A)** values. The next step is to signal the Detection Modules of these attacks to detect their attack signatures within the minute traffic, Figure 9.2.



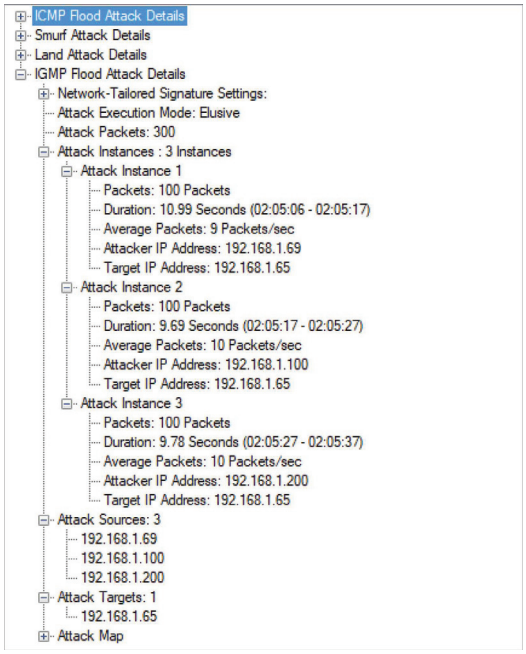
(a) ICMP Flood



(b) LAND



(c) Smurf



(d) IGMP Flood

Figure 9.2: Attack Detection Modules: Attack Minute 2 (NetDataCoP).

The Attack Detection Modules investigated the minute traffic and got the results depicted in Figure 9.2. ICMP Flood Detection Module, Figure 9.2a, was able to detect 1 ICMP Flood instance, executed using *50* packets and *45* connections, in a duration of *1.83* seconds, from attacker *'192.168.1.69'* to target *'192.168.1.68'*.

The LAND Detection Module, Figure 9.2b, detected 1 LAND instance, executed

using 49 packets, in a duration that elapsed for 3.02 seconds, towards target address *'192.168.1.65'*.

Also, the Smurf Detection Module, Figure 9.2c, detected 1 Smurf instance, executed using 45 packets and 30 connections, within a duration of 2.52 seconds, by 10 machines *(192.168.1.100 – 192.168.1.109)* to flood target *'192.168.1.65'*.

Finally, IGMP Flood Detection Module, Figure 9.2d, detected 3 IGMP Flood instances with 10 seconds time gap between the instances. IGMP Flood instances were executed by 3 different addresses to flood target *'192.168.1.65'*. Each instance included 100 packets and each instance lasted roughly for 10 seconds.

These results obtained by the attack modules conform with those inferred in Local Host-level Anomaly Detection, Table 9.9.

We wanted to demonstrate this minute specifically to introduce the concept of Distributed Denial of Service (or DDOS). Where multiple attackers coordinate there attacks to simultaneously execute several Denial of Service attacks against the same target. In our demonstration here, the intended DOS attacks were LAND, Smurf, and IGMP Flood attacks, all were executed concurrently to flood target *'192.168.1.65'*.

9.4 Comparative Analysis

In this section, we compare our work to the other work on Intrusion Detection systems. Most of Intrusion Detection research is either based on KDD or NSL-KDD dataset. Here, we use NSL-KDD because it is an improved version of KDD that addressed KDD's deficiencies. We compare the performance of our system's two stages to the results reported in two papers. Firstly, we compare our system's first stage – Anomaly Detection – to the results published in [2] (next section). Secondly, we compare the second stage – Signature Detection – to the

results from the paper [87], Section 9.4.2.

9.4.1 Anomaly Detection Performance

In [2], the authors used a hybrid intelligent approach for Intrusion Detection. Based on the authors' description, the word '*hybrid*' does not refer to application of different Intrusion Detection approaches, but to the usage of several different techniques. The framework of their proposed approach is depicted in Figure 9.3

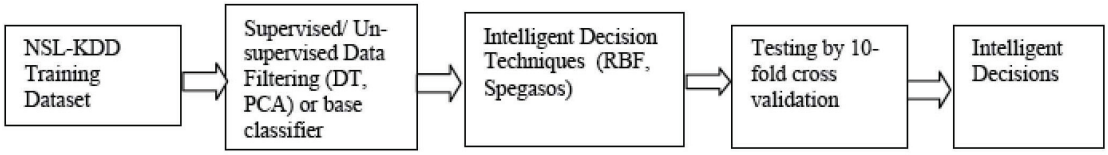


Figure 9.3: Intrusion Detection Framework proposed in [2].

The authors tried multiple combinations of the following approaches for intrusion detection:

1. Decision Trees (DT)
2. Principal Component Analysis(PCA)
3. Random Forest (RF)
4. Radial Basis Function Neural Network (RBF)
5. Stochastic variant of Piramol estimated sub-gradient solver in SVM (SPegasos)
6. Ensembles of Balanced Nested Dichotomies for Multi-class Problems (END)
7. Grading

They achieved the following best performance: 99.5% detection rate, and 0.1% false positive rate, using Random Forest with nested dichotomies and END. The comparison of our proposed Anomaly Detection approach to theirs is illustrated in Table 9.11 below.

Performance Matrices		'Published' Approach [2]	Our Approach
Detection Rate (%)	Normal	99.9	100.0
	Attack	99.5	99.4
False Positive Rate (%)	Normal	0.5	0.6
	Attack	0.1	0.0
F-Value (%)	Normal	99.8	100.0
	Attack	99.7	100.0
Precision (%)	Normal	99.6	99.5
	Attack	99.9	100.0
Recall (%)	Normal	99.9	100.0
	Attack	99.9	99.4

Table 9.11: Anomaly Performance Comparison

As it can be seen in the table, the application of our approach to NSL-KDD data shows a good performance. Except a few cases of lower values of performance matrices, the proposed ML-based system outperforms the approach presented in [2]. We obtained a perfect score for *F-Value* for both 'Normal' and 'Attack' classes.

9.4.2 Signature Detection Performance

Next, we compare our Signature Detection approach to that proposed in [87]. The main contribution of [87] is utilization of a dataset with the reduced number of features as an input to Naïve Bayes. The authors used three different features selection methods: Correlation-based Feature Selection (CFS), Information Gain (IG), and Gain Ratio (GR). It should be stated, that they did not classify individual attacks but categories of attacks: DOS, Probe, R2L and U2R. They proposed a new feature selection technique called Feature-Vitality Based Reduction Method (FVBRM). They used all four feature selection techniques to select features from the 41 features of NSL-KDD. They ended up having a dataset of 10 features using

CFS, a dataset with 14 features using GR, a dataset of 20 features using IG, and a last dataset of 24 features using FVBRM. They applied Naïve Bayes to all four reduced datasets and to the original NSL-KDD dataset. They showed that their proposed feature selection technique provides the best performance.

In our work, we select protocol-related features and build attack classsifiers based on the attack target matrix in Table 3.13. However, we have not perform any feature selection. We are comparing our approach to the one proposed in [87] based on classification of attack categories not individual attacks. Our methods take all features of NSL-KDD. The performance of our proposed Signature Detection approach compared to the results from [87] are demonstrated in Table 9.12 below.

Criteria	Paper Techniques					Our Approach
	CFS	GR	IG	FVBRM	NoReduction	
Attributes	10	14	20	24	41	41
Overall Accuracy (%)	97.55	95.30	95.21	97.78	95.11	99.94
DOS TPR (%)	96.0	90.6	93.0	98.7	93.5	100.0
Probe TPR (%)	97.0	96.0	98.6	98.8	97.8	99.8
R2L TPR (%)	94.0	94.2	95.9	96.1	96.8	95.9
U2R TPR (%)	4.0	4.0	20.0	64.0	56.0	61.5

Table 9.12: Signature Performance Comparison

For the case of Signature Detection, the proposed system behaves very well when compared with a number of approaches presented in [87]. The overall accuracy of the proposed approach is 99.94%, that is the highest value among all approaches presented in the table. If we look at individual types of attacks, only for two attacks, i.e., R2L and U2R, our system performs a bit worse. This happens for the approaches *FVBRM* and *NoReduction* in the case of R2L, and for *FVBRM* only in the case of U2R. The differences in the accuracy values are small.

Chapter 10

Conclusion and Future Work

In this thesis, we proposed, designed, and implemented an intelligent data-driven *Two-stage Hybrid Intrusion Detection System*. The hybrid nature of the system allows it to detect both anomalies and cyber attacks with high accuracy. Anomaly Detection, the first stage, allows the system to detect anomalous (not normal) network behaviors with high accuracy. Also, the system performs the high accuracy detection of signatures using the second stage: Signature Detection. To fulfil this task, the system consists of two modules: **Anomaly ID**, and **Signature ID**, together with a special module **NetDataCoP**.

NetDataCoP captures and processes network traffic using four sub-modules: *Traffic Capture*, *Packet Decipher*, *Connection Identification and Reconstruction*, and *Network Traffic Temporal Processing* sub-module. Traffic Capture sub-module captures network traffic in the form of stream of raw bytes. The captured stream of network traffic is processed by Packet Decipher sub-module, which uses socket technology and multi-threading technology to interpret the bytes of captured traffic streams and re-create corresponding protocol packets, along with their headers and fields. Connection Identification and Reconstruction sub-module identifies and reconstructs ICMP, TCP, and logical UDP connections via detailed examination of the reassembled packets. Finally,

the packets and connections are processed in a time-related manner to generate features over a specified time unit that describe the current state of network.

The first stage of the system, i.e., **Anomaly ID**, contains two sub-modules: Network-level Machine Learning-based Anomaly Detection and Host-level Threshold-based Anomaly Detection. The first of these sub-modules contains a number of classifiers constructed using different Machine Learning techniques. The sub-module takes the features describing the network traffic from the NetDataCoP as its input and passes them to the classifiers. The elements of Evidence Theory are used to aggregate the classifiers' outputs and determine a degree of confidence in the predicted state of the network. Once an anomaly is detected, the Host-level Threshold-based Anomaly ID processes the network traffic from a Local Host-level (local host-level) scope-centric perspective to extract detailed information about the detected anomaly.

The second stage in our system: **Signature ID**, also contains multiple sub-modules. One of them is Network-level Machine Learning-based Signature Detection while the other ones are *Attack Detection Modules*, each of them associated with a different attack. This time we focus on identification of a specific attack. The Network-level Machine Learning-based Signature Detection takes the features describing a network traffic over a time unit, one minute in our case, and using a number of classifiers constructed with Machine Learning methods and elements of Evidence Theory calculates the probabilistic outputs indicating degrees of confidence that specific attacks occurred. When one or more attacks are detected their corresponding Attack Detection Modules are invoked to search within the minute traffic for their respective network-tailored signatures. The findings, whether related to the detection of anomalies or cyber attacks, are reported to security administrators.

10.1 Conclusion

Over the course of the research, we worked on several supplementary but important research points. The ultimate accomplishment of our research is the multi-perspective feature-based network traffic description. It is a fundamental element that is used by other research tasks addressed in the thesis. The introduction of multiple perspectives, i.e., different points of view of 'looking at' a network traffic, and their inter-related features, makes our proposed network traffic description a very comprehensive one. The advantages of the description lie within its interoperability, where a value change of an individual feature leads to multiple perspective-specific effects.

The multi-perspective network description we proposed has led us to design and development of a thorough step-by-step analysis of attacks: starting with the detailed discussion of canonical attack signatures, and ending with attack signatures updated with the administrator knowledge. Further, the study of network anomaly profiles related to network topology has led us to network-tailored attack signatures, which proved to be more adequate for a given network. The network anomaly profile we used consists of 189 features. We created such a profile based on the proposed multi-perspective description of network traffic. The profile allows us and the developed system to draw a distinction between normal and anomalous behaviors of both network and individual hosts.

We are quite certain that the **NetDataCoP** is the first of its kind network analyzing system. It combines several functionalities. Besides capturing a network traffic, it is superior to other tools in performing a number of distinctive and important, from the point of view of intrusion detection, tasks: 1) it interprets all fields of protocol headers; 2) it performs a comprehensive analysis

of protocols of an application layer; 3) it identifies and reconstructs Internet (ICMP) and Transport (TCP and UDP) connections; and 4) finally, based on the deciphered packets and reconstructed connections, it is able to perform time-oriented processing of network traffic to generate features describing a current state of network over specified time intervals.

In our work, we propose the identification and reconstruction of logical connections of UDP-based applications. We found no other research work that tackles this in such details as we do. The identification and reconstruction of logical connections on top of UDP is a critical point to understand the behavior of UDP-based applications. For some application domains, like *Media Streaming*, this ability forms an important element in understanding how UDP applications communicate and behave.

We designed and built comprehensive data sets that include normal and attack network traffic. The data contains several cyber attacks executed in an elusive manner. We used this term to describe attacks that are performed in an non-standard way, i.e., attacks that differ from the canonical ones in several ways. For example, such attacks are executed over different, usually longer, periods of time, or involve different, usually, larger number of hosts. Such attacks represent a 'close to normal' network traffic, so it is difficult to identify them as attacks, or at the very least, as anomalies. We used these datasets in process of building and validating *Anomaly ID* and *Signature ID*.

In the domain of **Anomaly Detection**, we proposed *Global Network-level Machine Learning-based minute-wise Anomaly ID* approach. It aggregates classification results of multiple classifiers using elements of Evidence Theory (TBM) and provides better anomaly prediction results than that of a single classifier. As explained, different methods of combining belief masses, representing confidence levels in classifiers obtained with different datasets, has

been investigated. The simple averaging of obtained belief masses provided the best results. We have achieved high classification rate of 99.6 % accuracy against the evaluation dataset with only one anomalous point misclassified as normal. Even when we examine the probability values of predicting normal **BetP(N)** and anomalous **BetP(A)** conditions associated with the misclassified point: *0.5031* and *0.4969* respectively, we can notice that the values are nearly the same. This indicates that the predictions obtained from different classifiers are conflicting. Also, it indicates that the attack is elusive in nature, *i.e.*, it has created a network traffic very close to a normal one. This scenario can be an example of a real life intrusion situation. Also, within **Anomaly Detection**, we used the proposed network anomaly profile, in particular Local Host-level scope features, to pinpoint the origins of the detected anomalies via threshold-based comparison of calculated features describing individual host traffic against predefined threshold values representing normal traffic.

For **Signature Detection**, we proposed *Global Network-level Machine Learning-based minute-wise Signature Detection* approach. It also aggregates classification results of classifiers using elements of Evidence Theory (TBM) and provides better attack prediction results than that of a single classifier. Investigating different methods of combining belief masses did not present much difference in the classification performance as in the case of Anomaly ID. Using Bernoulli's Combination Rule, we achieved a high classification rate of 100% detection for existing attacks, with an accuracy of 99.6%. Only two data points representing 'Not-Attack' cases have been misclassified as 'Attack', and 544 data points have been classified correctly among 546 points of the evaluation dataset. The misclassification can be rationalized via similarity of attack behaviors, and existence of several attacks within the same data point.

Additionally, the developed *Attack Detection Modules* act as a safe guard against

misclassified attack instances. It means that an additional process of matching traffic patterns to the proposed network-tailored attack signatures can be treated as supplementary layer of verifying if an attack occured.

The final research outcome of this thesis is the *Intrusion Detection benchmark dataset*. The benchmark encloses our all datasets – Training, Test, and Evaluation. The benchmark can be considered the first of its kind intrusion detection dataset that represent cyber attacks in a temporal manner using multiple features. The dataset includes a number of realistic attack scenarios, *i.e.*, data points with multiple simultaneous attacks.

10.2 Future Work

In general, we would like to indicate that the contributions presented in this thesis can be treated as a initial work on new generation of intrusion detection systems able to continuously monitor network traffic and to deal on-line with more sophisticated types of attacks. The shown results are encouraging to perform more research in the area. Here, we highlight just a few future research tasks that relate to the presented work.

For the purpose of this thesis, we selected seven attacks belonging to DOS and Probe attack categories, further research can include more attacks from other categories.

The developed network anomaly profile is an implementation of features taken from the proposed multi-perspective network traffic description. We anticipate more work on new features of the multi-perspective description of network traffic, as well as on modifications of the existing ones. Therefore, a network anomaly profile will undergo modifications. We will investigate suitability of these changes to better define normal and anomalous states of

network traffic as well as a boundary between them.

For the developed NetDataCoP system, we envision multiple undertakings related to improvements of its modules:

Traffic Capture Module: currently, it captures only Ethernet II Frames with the emphasis on ARP and IPv4 packets, we will target other types of packets, *e.g.*, IPv6. and other Ethernet frames;

Packet Decipher Module: it analyzes the packet types explained in Section 5.2 and Appendix D, additional packet types will be interpreted;

Connection Identification and Reconstruction Module: the module identifies and reconstructs ICMP, TCP, and Logical UDP connections, extra research will focus on deduction of more connection features;

Network Traffic Temporal Processing Module: it performs a time-centric processing to generate 37 features describing the state of network over one minute interval, future research will include:

A. different sizes of Time Window: we will investigate the system's performance with a time window from one second to one hour, as well as with a sliding time window;

B. additional features: more features representing network traffic will be explore from the point of view of performance of the system.

Additional research can be performed on our approach used for identifying and reconstructing logical connections for UDP-based applications. It will occur in the form of investigating more UDP-based applications and identification of related connections.

In **Anomaly ID**, although the evidence theory has been proved to be an effective and interesting process of assigning and updating belief masses, more

future research can be done on the performance of the system. Here, we list a few most important possibilities:

Applying More Features: we will work on selection of more suitable features leading to the enhancement of predicability of TBM;

Utilizing Other Machine Learning Techniques: in our work, we used four Machine Learning and a Threshold-based Classifier, namely; J48 Decision Tree, Support Vector Machine, Logistic Regression, K-Means Clustering, and MAX-based classifier; we envision more work on combining variety of classifiers built with different Machine Learning techniques.

Furthermore, we will look at additional host-centric features for the *Local Host-level Threshold-based Anomaly ID*. Our goal will be to design, implement, and apply new features to ensure a more detailed analysis of the behavior of individual hosts.

In the case of **Signature ID**, we proposed *Global Network-level Machine Learning-based Signature ID* approach that is based on seven attacks and three Machine Learning classifiers: J48 Decision Tree, Support Vector Machine, and Simple Logistic. In our work, we tried several combinations of features to be used with the classifiers, and the best results were reported in this thesis. We anticipate that for other attacks, other Machine Learning classifiers could provide good results. The investigations of multiple combinations of features and classifiers is a very interesting future undertaking. The *Attack Detection Modules* that analyze a network traffic data to extract specific information regarding a given attack can be further enhanced. The activities leading to such enhancements would include more work on original as well as on already updated by our work multiple attack signatures.

Intrusion Detection benchmark datasets can be also further improved and

extended. We plan to design and build at least two more benchmark datasets: one pertaining a connection-based format; and the other one representing data records in a packet-based format.

List of References

[1] B.A. Forouzan and S.C. Fegan. *Data Communications and Networking*. McGraw-Hill Forouzan networking series. McGraw-Hill Higher Education, 2007. ISBN 9780072967753. URL https://books.google.ca/books?id=bwUNZvJbEeQC.

[2] Mrutyunjaya Panda, Ajith Abraham, and Manas Ranjan Patra. A hybrid intelligent approach for network intrusion detection. *Procedia Engineering*, 30:1 – 9, 2012. ISSN 1877-7058. URL http://www.sciencedirect.com/science/article/pii/S1877705812008375. International Conference on Communication Technology and System Design 2011.

[3] Wikipedia. Ethernet frame, 2017. https://en.wikipedia.org/wiki/Ethernet_frame.

[4] Cisco. One Byte at a Time: Bootstrapping with BOOTP and DHCP - The Internet Protocol Journal - Volume 5, Number 2, 2002. http://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-22/dhcp.html.

[5] P. Mockapetris. Domain names: Implementation specification. RFC 883, November 1983. URL https://rfc-editor.org/rfc/rfc883.txt.

[6] Dr. Levon Esibov, Dave Thaler, and Dr. Bernard D. Aboba Ph.D. Link-local Multicast Name Resolution (LLMNR). RFC 4795, January 2007. URL https://rfc-editor.org/rfc/rfc4795.txt.

[7] Protocol standard for a NetBIOS service on a TCP/UDP transport: Detailed specifications. RFC 1002, March 1987. URL https://rfc-editor.org/rfc/rfc1002.txt.

[8] Douglas Bruey. SNMP: Simple Network Management Protocol, 2005. http://www.rane.com/note161.html.

[9] Alan Presser et al. UPnP™ Device Architecture 1.1, 2008. www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.0-20080424.pdf.

[10] Dhruba Kumar Bhattacharyya and Jugal Kumar Kalita. *Network Anomaly Detection: A Machine Learning Perspective*. Chapman & Hall/CRC, 2013. ISBN 1466582081, 9781466582088.

[11] Karen A. Scarfone and Peter M. Mell. Sp 800-94. guide to intrusion detection and prevention systems (IDPS). Technical report, National Institute of Standards & Technology, Gaithersburg, MD, United States, 2007.

[12] Wikipedia. Intrusion Detection System, 2017. https://en.wikipedia.org/wiki/Intrusion_detection_system.

[13] Kristopher Kendall. A database of computer attacks for the evaluation of intrusion detection systems. Masters thesis, Massachusetts Institute of Technology, Cambridge, MA, 1999.

[14] Kristopher Kendall. 1999 DARPA Intrusion Detection Evaluation Data Set. https://www.ll.mit.edu/ideval/data/1999data.html, 1999. [Online; accessed 06-April-2017].

[15] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. A detailed analysis of the kdd cup 99 data set. In *IEEE Symposium on Computational Intelligence for Security and Defense Applications, 2009. CISDA 2009.*, pages 1–6. IEEE, 2009.

[16] Preeti Aggarwal and Sudhir Kumar Sharma. Analysis of kdd dataset attributes-class wise for intrusion detection. *Procedia Computer Science*, 57: 842–851, 2015.

[17] Prasanta Gogoi, Monowar H Bhuyan, DK Bhattacharyya, and Jugal K Kalita. Packet and flow based network intrusion dataset. In *International Conference on Contemporary Computing*, pages 322–334. Springer, 2012.

[18] Massachusetts Institute of Technology Lincolin Laboratory. Kdd cup 1999, 2014. http://kdd.ics.uci.edu/databases/kddcup99/.

[19] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

[20] Johannes Fürnkranz. *Decision Lists and Decision Trees*, pages 261–262. Springer US, Boston, MA, 2010. ISBN 978-0-387-30164-8.

[21] Sumeet Dua and Xian Du. *Data mining and machine learning in cybersecurity*. CRC press, 2016.

[22] Claude Sammut and Geoffrey I. Webb, editors. *Logistic Regression*, pages 631–631. Springer US, Boston, MA, 2010. ISBN 978-0-387-30164-8. doi: 10.1007/978-0-387-30164-8_493. URL http://dx.doi.org/10.1007/978-0-387-30164-8_493.

[23] Scott Menard. *Logistic Regression: From Introductory to Advanced Concepts and Applications*. SAGE Publications, Inc., Thousand Oaks, California, 2010. doi: 10.4135/9781483348964.

[24] Xin Jin and Jiawei Han. *K-Means Clustering*, pages 563–564. Springer US, Boston, MA, 2010. ISBN 978-0-387-30164-8. doi: 10.1007/978-0-387-30164-8_425. URL http://dx.doi.org/10.1007/978-0-387-30164-8_425.

[25] Glenn Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, Princeton, 1976.

[26] Lotfi A. Zadeh. A simple view of the dempster-shafer theory of evidence and its implication for the rule of combination. *Artificial Intelligence Magazine*, 7(2):85–90, 1986.

[27] Philippe Smets. The concept of distinct evidence. In *IPMU 92 Proceedings. pg*, pages 789–794, 1992.

[28] Marek Reformat, Petr Musilek, and Efe Igbide. Intelligent analysis of software maintenance data. In *Advances in Machine Learning Applications in Software Engineering*, pages 14–51. IGI Global, 2007.

[29] Mueen Uddin, Azizah Abdul Rehman, Naeem Uddin, Jamshed Memon, Raed Alsaqour, and Suhail Kazi. Signature-based multi-layer distributed intrusion detection system using mobile agents. *International Journal of Network Security*, 15(2):97–105, 3 2013. ISSN 1816-353X.

[30] R. Chetan and D. V. Ashoka. Data mining based network intrusion detection system: A database centric approach. In *2012 International Conference on Computer Communication and Informatics*, pages 1–6, Jan 2012. doi: 10.1109/ICCCI.2012.6158816.

[31] C. V. Raman and Atul Negi. A hybrid method to intrusion detection systems using hmm. In *Proceedings of the Second International Conference on Distributed Computing and Internet Technology*, ICDCIT'05, pages 389–396, Berlin, Heidelberg, 2005. Springer-Verlag. ISBN 3-540-30999-3, 978-3-540-30999-4. doi: 10.1007/11604655_44. URL http://dx.doi.org/10.1007/11604655_44.

[32] Mahsa Khosronejad, Elham Sharififar, Hasan Ahmadi Torshizi, and Mehrdad Jalali. Developing a hybrid method of hidden markov models and c5.0 as a intrusion detection system. *International Journal of Database Theory and Application*, 6(5):165–174, 2013. ISSN 1816-353X.

[33] M. Doroudian, N. Arastouie, M. Talebi, and A. R. Ghanbarian. Multilayered database intrusion detection system for detecting malicious behaviors in big data transaction. In *2015 Second International Conference on Information Security and Cyber Forensics (InfoSec)*, pages 105–110, Nov 2015.

[34] Mingjun Wei, Lichun Xia, and Jingjing Su. *Research on the Application of Improved K-Means in Intrusion Detection*, pages 673–678. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011. ISBN 978-3-642-27503-6. doi: 10.1007/978-3-642-27503-6_92. URL http://dx.doi.org/10.1007/978-3-642-27503-6_92.

[35] P. Satam. Cross layer anomaly based intrusion detection system. In *2015 IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops*, pages 157–161, Sept 2015. doi: 10.1109/SASOW.2015.31.

[36] Suvasini Panigrahi, Shamik Sural, and Arun K. Majumdar. Two-stage database intrusion detection by combining multiple evidence and belief update. *Information Systems Frontiers*, 15(1):35–53, 2013. ISSN 1572-9419.

[37] Gianluigi Folino, Francesco Sergio Pisani, and Pietro Sabatino. A distributed intrusion detection framework based on evolved specialized ensembles of classifiers. In *European Conference on the Applications of Evolutionary Computation*, pages 315–331. Springer, 2016.

[38] Nour Moustafa and Jill Slay. A hybrid feature selection for network intrusion detection systems: Central points. In *the 16th Australian Information Warfare Conference*, pages 5–13. Security Research Institute, Edith Cowan University, 2015.

[39] Jasmin Kevric, Samed Jukic, and Abdulhamit Subasi. An effective combining classifier approach using tree algorithms for network intrusion detection. *Neural Computing and Applications*, pages 1–8, 2016.

[40] Nutan Farah Haq, Abdur Rahman Onik, and Faisal Muhammad Shah. An ensemble framework of anomaly detection using hybridized feature selection approach (hfsa). In *SAI Intelligent Systems Conference (IntelliSys), 2015*, pages 989–995. IEEE, 2015.

[41] Kathleen Goeschel. Reducing false positives in intrusion detection systems using data-mining techniques utilizing support vector machines, decision trees, and naive bayes for off-line analysis. In *SoutheastCon, 2016*, pages 1–6. IEEE, 2016.

[42] Vikram Kumaran. Event stream database based architecture to detect network intrusion: (industry article). In *Proceedings of the 7th ACM International Conference on Distributed Event-based Systems*, DEBS '13, pages 241–248, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1758-0. doi: 10.1145/2488222.2488276. URL http://doi.acm.org/10.1145/2488222.2488276.

[43] John 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 (TISSEC)*, 3(4):262–294, 2000.

[44] Gerald Combs. Wireshark, 2017. https://www.wireshark.org/.

[45] Cisco. Snort, 2017. https://www.snort.org/.

[46] Riverbed. Winpcap, 2013. https://www.winpcap.org/.

[47] GitHub. Pcap.net, 2017. https://github.com/PcapDotNet/Pcap.Net.

[48] D. Plummer. Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware. RFC 826 (Standard), November 1982. URL http://www.ietf.org/rfc/rfc826.txt. Updated by RFCs 5227, 5494.

[49] S. Cheshire, B. Aboba, and E. Guttman. Dynamic Configuration of IPv4 Link-Local Addresses. RFC 3927 (Proposed Standard), May 2005. URL http://www.ietf.org/rfc/rfc3927.txt.

[50] S. Cheshire. IPv4 Address Conflict Detection. RFC 5227 (Proposed Standard), July 2008. URL http://www.ietf.org/rfc/rfc5227.txt.

[51] J. Doyle and J.D.H. Carroll. *Routing TCP/IP*. Number v. 1 in CCIE Professional Development Routing TCP/IP. Cisco Press, 2005. ISBN 9781587052026. URL https://books.google.ca/books?id=JjdF2yWqJAwC.

[52] *RFC 791 Internet Protocol - DARPA Inernet Programm, Protocol Specification*. Internet Engineering Task Force, September 1981.

[53] S. E. Deering. Host extensions for IP multicasting. RFC 988, jul 1986. URL https://rfc-editor.org/rfc/rfc988.txt.

[54] S. E. Deering. Host extensions for IP multicasting. RFC 1112, aug 1989. URL https://rfc-editor.org/rfc/rfc1112.txt.

[55] J. Postel. Internet Control Message Protocol. RFC 792, sep 1981.

[56] Bill Fenner. Internet Group Management Protocol, Version 2. RFC 2236, nov 1997.

[57] Haixiang He, Brian Haberman, and Hal Sandick. Internet Group Management Protocol (IGMP) / Multicast Listener Discovery (MLD)-Based Multicast Forwarding ("IGMP/MLD Proxying"). RFC 4605, aug 2006.

[58] J. Postel. Transmission Control Protocol. RFC 793, sep 1981.

[59] Robert T. Braden. Requirements for Internet Hosts - Communication Layers. RFC 1122, October 1989. URL https://rfc-editor.org/rfc/rfc1122.txt.

[60] J. Postel. User Datagram Protocol. RFC 768, aug 1980. URL https://rfc-editor.org/rfc/rfc768.txt.

[61] John Gilmore Bill Croft. Bootstrap Protocol. RFC 951, September 1985. URL https://rfc-editor.org/rfc/rfc951.txt.

[62] Walter Wimer. Clarifications and Extensions for the Bootstrap Protocol. RFC 1532, October 1993. URL https://rfc-editor.org/rfc/rfc1532.txt.

[63] Ralph Droms. Interoperation Between DHCP and BOOTP. RFC 1534, October 1993. URL https://rfc-editor.org/rfc/rfc1534.txt.

[64] Walter Wimer. Clarifications and Extensions for the Bootstrap Protocol. RFC 1542, October 1993. URL https://rfc-editor.org/rfc/rfc1542.txt.

[65] Joyce K. Reynolds. BOOTP Vendor Information Extensions. RFC 1497, August 1993. URL https://rfc-editor.org/rfc/rfc1497.txt.

[66] P. Mockapetris. Domain names - implementation and specification. RFC 1035, November 1987. URL https://rfc-editor.org/rfc/rfc1035.txt.

[67] Vladimir Ksinant, Christian Huitema, Dr. Susan Thomson, and Mohsen Souissi. DNS Extensions to Support IP Version 6. RFC 3596, October 2003. URL https://rfc-editor.org/rfc/rfc3596.txt.

[68] Ray Bellis. DNS Transport over TCP - Implementation Requirements. RFC 5966, August 2010. URL https://rfc-editor.org/rfc/rfc5966.txt.

[69] Remco Mook and Bert Hubert. Measures for Making DNS More Resilient against Forged Answers. RFC 5452, January 2009. URL https://rfc-editor.org/rfc/rfc5452.txt.

[70] Protocol standard for a NetBIOS service on a TCP/UDP transport: Concepts and methods. RFC 1001, March 1987. URL https://rfc-editor.org/rfc/rfc1001.txt.

[71] Dr. Jeff D. Case, Mark Fedor, James R. Davin, and Martin Lee Schoffstall. Simple Network Management Protocol (SNMP). RFC 1098, April 1989. URL https://rfc-editor.org/rfc/rfc1098.txt.

[72] Mark Fedor, James R. Davin, Martin Lee Schoffstall, and Dr. Jeff D. Case. Simple Network Management Protocol (SNMP). RFC 1157, May 1990. URL https://rfc-editor.org/rfc/rfc1157.txt.

[73] Simple Service Discovery Protocol/1.0, 1999. https://tools.ietf.org/html/draft-cai-ssdp-v1-03.

[74] SSDP: Networked Home Entertainment Devices (NHED) Extensions, 2015. https://winprotocoldoc.blob.core.windows.net/.../MS-SSDP/[MS-SSDP].pdf.

[75] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC 3550 (Standard), July 2003. URL http://www.ietf.org/rfc/rfc3550.txt. Updated by RFC 5506.

[76] H. Schulzrinne and S. Casner. RTP Profile for Audio and Video Conferences with Minimal Control. RFC 3551 (Standard), July 2003. URL http://www.ietf.org/rfc/rfc3551.txt.

[77] D. Hoffman, G. Fernando, V. Goyal, and M. Civanlar. RTP Payload Format for MPEG1/MPEG2 Video. RFC 2250 (Proposed Standard), January 1998. URL http://www.ietf.org/rfc/rfc2250.txt.

[78] SourceForge.net. Hyenae download, 2017. https://sourceforge.net/projects/hyenae/.

[79] ostinato.org. Ostinato network traffic generator, 2017. http://ostinato.org/.

[80] solarwinds.com. Solarwinds engineer's toolset: Ping sweep, 2017. http://www.solarwinds.com/engineers-toolset/ping-sweep.

[81] angryip.org. Angry ip scanner, 2017. http://angryip.org/download/#windows.

[82] advanced-ip scanner.com. Advanced ip scanner, 2017. https://www.advanced-ip-scanner.com/.

[83] solarwinds.com. Solarwinds engineer's toolset: Dns audit, 2017. http://www.solarwinds.com/engineers-toolset/dns-audit.

[84] solarwinds.com. Solarwinds engineer's toolset: Dns who is resolver, 2017. http://www.solarwinds.com/engineers-toolset/dns-whois-resolver.

[85] Nir Sofer. Netbscanner v1.11, 2016. https://www.nirsoft.net/utils/netbios_scanner.html.

[86] nmap.org. Nmap (network mapper), 2016. https://nmap.org/.

[87] Saurabh Mukherjee and Neelam Sharma. Intrusion detection using naive bayes classifier with feature reduction. *Procedia Technology*, 4: 119 – 128, 2012. ISSN 2212-0173. URL http://www.sciencedirect.com/science/article/pii/S2212017312002964. 2nd International Conference on Computer, Communication, Control and Information Technology(C3IT-2012) on February 25 - 26, 2012.

[88] Michelle S. Cotton, Lars Eggert, Dr. Joseph D. Touch, Magnus Westerlund, and Stuart Cheshire. Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry. RFC 6335, aug 2011. URL https://rfc-editor.org/rfc/rfc6335.txt.

[89] Dr. Marshall T. Rose and Keith McCloghrie. Management Information Base for Network Management of TCP/IP-based internets: MIB-II. RFC 1213, mar 1991. URL https://rfc-editor.org/rfc/rfc1213.txt.

Appendix A

Network Anomaly Profile

In this appendix, we explain how we designed and implemented our network anomaly profile. All the provided explanation is based on the sessions that are chosen to create the profile. We calculate features whether they are time-dependent or time-independent features, all of them are designed based on the proposed multi-perspective feature-based description of network traffic.

A.1 Profile Features

In this section we are going to provide the features of our proposed network anomaly profile, grouped by protocol.

A.1.1 ARP Features

ID	Layer: **Network Interface** Protocol: **ARP**			
	Scope	Auxiliary Perspectives	Feature Name	Feature Value (threshold)
1	Network	Packet Temporal	Packet Count	[350,2406]
2			Packets Per Second	0.4
3			Packets Per Minute	24.2
4			Packets Per Hour	1451.3
5			Request Packets	[235,1328]
6			Requests Per Second	0.2
7			Requests Per Minute	12.1
8			Requests Per Hour	723.9
9			Response Packets	[115,1135]
10			Responses Per Second	0.2
11			Responses Per Minute	12.1
12			Responses Per Hour	726.8
13		Address	Source Addresses	7
14			Destination Addresses	8
15			Request Source Addresses	5
16			Request Destination Addresses	8
17			Response Source Addresses	6
18			Response Destination Addresses	4
19	Host	Address	Destinations Per Source	[1,6]
20		Packet	Requests Per Destinations	[1,365]
21		Packet	Overall Requests Per Source	[1,892]
22		Packet	Responses Per Sources	[1,353]
23		Packet	Overall Responses Per Destination	[4,582]

Table A.1: ARP Protocol Features

Packet Count Range of collected ARP packets issued within sessions.

Packets Per Second The maximum average rate of ARP packets per second.

Packets Per Minute The maximum average rate of ARP packets per minute.

Packets Per Hour The maximum average rate of ARP packets per hour.

Request Packets Range of collected ARP request packets issued within sessions.

Requests Per Second The maximum average ARP request rate per second.

Requests Per Minute The maximum average ARP request rate per minute.

Requests Per Hour The maximum average ARP request rate per hour.

Response Packets Range of collected ARP response packets issued within sessions.

Responses Per Second The maximum average ARP response rate per second.

Responses Per Minute The maximum average ARP response rate per minute.

Responses Per Hour The maximum average ARP response rate per hour.

Source Addresses The maximum distinct ARP source address count.

Destination Addresses The maximum distinct ARP destination address count.

Request Source Addresses The maximum distinct ARP request source address count.

Request Destination Addresses The maximum distinct ARP request destination address count

Response Source Addresses The maximum distinct ARP response source address count

Response Destination Addresses The maximum distinct ARP response destination address count.

Destinations Per Source The maximum distinct ARP destinations contacted by a certain source.

Requests Per Destinations The maximum number of ARP requests issued by a source to a destination.

Overall Requests Per Source The maximum number of overall ARP requests issued by a certain source.

Responses Per Sources The maximum number of ARP response packets received by a destination from a source.

Overall Responses Per Destination The maximum number of ARP response packets received by a destination.

A.1.2 ICMP Features

In this section we illustrate and explain ICMP protocol features, Table A.2.

ID	Layer: **Internet** Protocol: **ICMP**			
	Scope	Auxiliary Perspectives	Feature Name	Feature Value (threshold)
1	Network	Packet Temporal	Packet Count	[0,425]
2			Packets Per Second	0
3			Packets Per Minute	3.2
4			Packets Per Hour	189.6
5			Error Packets	[0,417]
6			Error Per Second	0
7			Error Per Minute	3.1
8			Error Per Hour	188
9			Query Packets	[0,8]
10			Query Packets Per Second	0
11			Query Packets Per Minute	0
12			Query Packets Per Hour	1.6
13			Query Request Packets	[0,4]
14			Query Request Packets Per Second	0
15			Query Request Packets Per Minute	0
16			Query Request Packets Per Hour	0.8
17			Query Response Packets	[0,4]
18			Query Response Packets Per Second	0
19			Query Response Packets Per Minute	0
20			Query Response Packets Per Hour	0.8
21		Address	Source Addresses	2
22			Destination Addresses	2
23			Request Source Addresses	1
24			Request Destination Addresses	1
25			Response Source Addresses	1
26			Response Destination Addresses	1
27		Packet	Maximum Packet Size	84
28		Connection	Connection Count	[0,2]
29		Connection Temporal	Connections Per Second	0
30			Connections Per Minute	0
31			Connections Per Hour	0.2
32	Host	Address	Destinations Per Source	[0,1]
33		Packet	Requests Per Destinations	[0,4]
34			Overall Requests Per Source	[0,4]
35			Responses Per Sources	[0,4]
36			Overall Responses Per Destination	[0,4]
37		Connection	Connections Per Destination	[0,2]
38			Overall Connections Per Source	[0,2]

Table A.2: ICMP Protocol Features

Packet Count Range of collected ICMP packets.

Packets Per Second The maximum average ICMP packet rate per second.

Packets Per Minute The maximum average ICMP packets rate per minute.

Packets Per Hour The maximum average ICMP packets rate per hour.

Error Packets Range of collected ICMP error packets.

Error Packets Per Second The maximum average ICMP error packets rate per second.

Error Packets Per Minute The maximum average ICMP error packets rate per minute.

Error Packets Per Hour The maximum average ICMP error packets rate per hour.

Query Packets Range of collected ICMP query packets.

Query Packets Per Second The maximum average ICMP query packets rate per second.

Query Packets Per Minute The maximum average ICMP query packets rate per minute.

Query Packets Per Hour The maximum average ICMP query packets rate per hour.

Query Request Packets Range of collected ICMP query request packets.

Query Request Packets Per Second The maximum average ICMP query request packets rate per second.

Query Request Packets Per Minute The maximum average ICMP query request packets rate per minute.

Query Request Packets Per Hour The maximum average ICMP query request packets rate per hour.

Query Response Packets Range of collected ICMP query response packets.

Query Response Packets Per Second The maximum average ICMP query response packets rate per second.

Query Response Packets Per Minute The maximum average ICMP query response packets rate per minute.

Query Response Packets Per Hour The maximum average ICMP query response packets rate per hour.

Source Addresses The maximum overall number of distinct ICMP source addresses.

Destination Addresses The maximum overall number of distinct ICMP destination addresses.

Request Source Addresses The maximum overall number of distinct ICMP request source addresses.

Request Destination Addresses The maximum overall number of distinct ICMP request destination addresses.

Response Source Addresses The maximum overall number of distinct ICMP response source addresses.

Response Destination Addresses The maximum overall number of distinct ICMP response destination addresses.

Maximum Packet Size The Maximum ICMP packet size.

Connection Count Range of collected ICMP connections.

Connections Per Second The maximum average rate of ICMP connections per second.

Connections Per Minute The maximum average rate of ICMP connections per minute.

Connections Per Hour The maximum average rate of ICMP connections per hour.

Destinations Per Source The maximum number of distinct ICMP destinations contacted by a source.

Requests Per Destinations The maximum number of ICMP requests issued by a certain source.

Overall Requests Per Source The maximum number of overall ICMP requests issued by a source.

Responses Per Sources The maximum number of ICMP responses received by a destination from a source.

Overall Responses Per Destination The maximum overall number of ICMP response packets received by a destination.

Connections Per Destination The maximum number of ICMP connections issued by a source to a destination.

Overall Connections Per Source The maximum number of overall ICMP connections issued by a certain source.

A.1.3 IGMP Features

ID	Layer: **Internet** Protocol: **IGMP**			
	Scope	Auxiliary Perspectives	Feature Name	Feature Value (threshold)
1	Network	Packet Temporal	Packet Count	[443,1189]
2			Packets Per Second	0.3
3			Packets Per Minute	16.9
4			Packets Per Hour	1011.6
5			Multicast Packets	[443,1189]
6			Multicast Packets Per Second	0.3
7			Multicast Packets Per Minute	16.9
8			Multicast Packets Per Hour	1011.6
9			Unicast Packets	[0,0]
10			Unicast Packets Per Second	0
11			Unicast Packets Per Minute	0
12			Unicast Packets Per Hour	0
13	Host		Multicast Packets Per Second	[0,0.1]
14			Unicast Packets Per Second	0

Table A.3: IGMP Protocol Features

Packet Count Range of collected IGMP packets.

Packets Per Second The maximum average IGMP packet rate per second.

Packets Per Minute The maximum average IGMP packets rate per minute.

Packets Per Hour The maximum average IGMP packets rate per hour.

Packet Count Range of collected Multicast IGMP packets.

Packets Per Second The maximum average Multicast IGMP packet rate per second.

Packets Per Minute The maximum average Multicast IGMP packets rate per minute.

Packets Per Hour The maximum average Multicast IGMP packets rate per hour.

Packet Count Range of collected Unicast IGMP packets.

Packets Per Second The maximum average Unicast IGMP packet rate per second.

Packets Per Minute The maximum average Unicast IGMP packets rate per minute.

Packets Per Hour The maximum average Unicast IGMP packets rate per hour.

Multicast Packets Per Second The maximum average Multicast IGMP packets issued by a source per second.

Unicast Packets Per Second The maximum average Unicast IGMP packets issued by a source per second.

A.1.4 Network Interface Layer Features

ID	Layer: **Internet** Protocol: **General**			
	Scope	Auxiliary Perspectives	Feature Name	Feature Value (threshold)
1	Network	Packet Address	LAND Indicator	0

Table A.4: Network Interface Layer Features

LAND Indicator An indicator that signifies the existence of LAND attack.

A.1.5 TCP Features

ID	Layer: **Transport** Protocol: **TCP**			
	Scope	Auxiliary Perspectives	Feature Name	Feature Value (threshold)
1	Network	Packet Temporal	Packet Count	[4084,35105]
2			Packets Per Second	3.7
3			Packets Per Minute	222.8
4			Packets Per Hour	13371.6
5		Connection Temporal	Connection Count	[79,1809]
6			Connections Per Second	0.2
7			Connections Per Minute	9
8			Connections Per Hour	538.8
9		Protocol	Invoked Services	[3,7]
10		Protocol,Packet	Service Packets	[1,28189]
11	Host	Protocol Packet Connection	Invoked Services	[1,5]
12			Exchanged Packets	[284,18874]
13			Established Connections	[8,56]

Table A.5: TCP Protocol Features

Packet Count Range of collected TCP packets.

Packets Per Second The maximum average TCP packet rate per second.

Packets Per Minute The maximum average TCP packets rate per minute.

Packets Per Hour The maximum average TCP packets rate per hour.

Packet Count Range of collected TCP Connections.

Packets Per Second The maximum average TCP Connections rate per second.

Packets Per Minute The maximum average TCP Connections rate per minute.

Packets Per Hour The maximum average TCP Connections rate per hour.

Invoked Services Range of overall invoked TCP services.

Service Packets Range of packets for invoked TCP services.

Host Invoked Services Range of invoked TCP services between 2 hosts.

Host Exchanged Packets Range of packets for invoked TCP services between 2 hosts.

Host Established Connections Range of TCP connections established between 2 hosts.

A.1.6 UDP Features

ID	Layer: **Transport** Protocol: **UDP**			
	Scope	Auxiliary Perspectives	Feature Name	Feature Value (threshold)
1	Network	Packet Temporal	Packet Count	[6987,10640]
2			Packets Per Second	2.2
3			Packets Per Minute	130.1
4			Packets Per Hour	7806.2
5		Connection Temporal	Connection Count	[104,433]
6			Connections Per Second	0.1
7			Connections Per Minute	3.6
8			Connections Per Hour	213.9
9		Protocol	Invoked Services	[1,7]
10		Protocol,Packet	Service Packets	[1,4066]
11	Host	Protocol	Invoked Services	[1,2]
12		Packet	Exchanged Packets	[132,534]
13		Connection	Established Connections	[31,144]

Table A.6: UDP Protocol Features

Packet Count Range of collected UDP packets.

Packets Per Second The maximum average UDP packet rate per second.

Packets Per Minute The maximum average UDP packets rate per minute.

Packets Per Hour The maximum average UDP packets rate per hour.

Packet Count Range of collected UDP Connections.

Packets Per Second The maximum average UDP Connections rate per second.

Packets Per Minute The maximum average UDP Connections rate per minute.

Packets Per Hour The maximum average UDP Connections rate per hour.

Invoked Services Range of overall invoked UDP services.

Service Packets Range of packets for invoked UDP services.

Host Invoked Services Range of invoked UDP services between 2 hosts.

Host Exchanged Packets Range of packets for invoked UDP services between 2 hosts.

Host Established Connections Range of UDP connections established between 2 hosts.

A.1.7 Transport Layer Features

ID	Layer: **Transport** Protocol: **General**			
	Scope	Auxiliary Perspectives	Feature Name	Feature Value (threshold)
1	Host	Protocol	Invoked Transport Services	[2,7]
2		Packet Protocol	Exchanged Transport Packets	[416,19356]
3		Packet Connection	Established Transport Connections	[87,185]
4		Protocol	Attempted Transport Services	[0,2]
5		Packet Protocol	Exchanged Attempted Transport Packets	[0,94]
6		Packet Connection	Established Attempted Transport Connections	[0,16]

Table A.7: Transport Layer Features

Invoked Transport Services Range of overall invoked Transport services.

Exchanged Transport Packets Range of overall exchanged Transport packets for invoked services.

Established Transport Connections Range of overall established Transport connections.

Attempted Transport Services Range of overall attempted Transport services.

Exchanged Attempted Transport Packets Range of overall exchanged Transport packets for attempted services.

Established Attempted Transport Connections Range of overall attempted Transport connections.

A.1.8 DNS Features

ID	Layer: **Application** Protocol: **DNS**			
	Scope	Auxiliary Perspectives	Feature Name	Feature Value (threshold)
1	Network	Packet	Packet Count	[0,417]
2		Packet Temporal	Packets Per Second	0
3			Packets Per Minute	1.6
4			Packets Per Hour	98.1
5		Address	Multicast Destinations	[0,0]
6			Unicast Destinations	[0,4]
7	Host	Protocol	Name Resolution Indicator	0
8		Address	Destinations Per Source	[0,2]

Table A.8: DNS Protocol Features

Packet Count Range of collected DNS packets.

Packets Per Second The maximum average DNS packet rate per second.

Packets Per Minute The maximum average DNS packets rate per minute.

Packets Per Hour The maximum average DNS packets rate per hour.

Multicast Destinations Range of Multicast DNS destinations.

Unicast Destinations Range of Unicast DNS destinations.

Name Resolution Indicator An indicator that signifies the existence of DNS address to name resolution.

Destinations Per Source Range of DNS destinations contacted by a source.

A.1.9 NBNS Features

ID	Layer: **Application** Protocol: **DNS**			
	Scope	Auxiliary Perspectives	Feature Name	Feature Value (threshold)
1	Network	Packet	Packet Count	[0,125]
2		Packet Temporal	Packets Per Second	0
3			Packets Per Minute	0.8
4			Packets Per Hour	48.7
5		Address	Multicast Destinations	[0,1]
6			Unicast Destinations	[0,1]
7	Host	Protocol	Name Resolution Indicator	0
8		Address	Destinations Per Source	[0,2]

Table A.9: NBNS Protocol Features

Packet Count Range of collected NBNS packets.

Packets Per Second The maximum average NBNS packet rate per second.

Packets Per Minute The maximum average NBNS packets rate per minute.

Packets Per Hour The maximum average NBNS packets rate per hour.

Multicast Destinations Range of Multicast NBNS destinations.

Unicast Destinations Range of Unicast NBNS destinations.

Name Resolution Indicator An indicator that signifies the existence of NBNS address to name resolution.

Destinations Per Source Range of NBNS destinations contacted by a source.

A.1.10 LLMNR Features

ID	Layer: **Application** Protocol: **DNS**			
	Scope	Auxiliary Perspectives	Feature Name	Feature Value (threshold)
1	Network	Packet	Packet Count	[0,74]
2		Packet Temporal	Packets Per Second	0
3			Packets Per Minute	0.6
4			Packets Per Hour	34.6
5		Address	Multicast Destinations	[0,1]
6			Unicast Destinations	[0,0]
7	Host	Protocol	Name Resolution Indicator	0
8		Address	Destinations Per Source	[0,1]

Table A.10: LLMNR Protocol Features

Packet Count Range of collected LLMNR packets.

Packets Per Second The maximum average LLMNR packet rate per second.

Packets Per Minute The maximum average LLMNR packets rate per minute.

Packets Per Hour The maximum average LLMNR packets rate per hour.

Multicast Destinations Range of Multicast LLMNR destinations.

Unicast Destinations Range of Unicast LLMNR destinations.

Name Resolution Indicator An indicator that signifies the existence of LLMNR address to name resolution.

Destinations Per Source Range of LLMNR destinations contacted by a source.

A.2 Utilization in Intrusion Detection

We are going demonstrate how to utilize our implemented network profile within Anomaly Detection. We perform Anomaly ID by examining Global Network-level scope-centric features, in order to detect global evidences of anomalies. In the demonstration here we use 2 example packet capture sessions; Session 'A'

and Session 'B'. Session 'A' contains IPSweep attack while Session 'B' includes ICMP Flood Attack, so it is suffice to use only ARP and ICMP features for this demonstration. Table A.11 demonstrates ARP and ICMP Global Network-level features and the corresponding feature values from Sessions 'A' and 'B'.

ID	Layer: **Network Interface** Protocol: **ARP** Scope: **Global Network-level**		Example Sessions	
	Feature Name	Feature Value	Session 'A'	Session 'B'
1	Packet Count	[350,2406]	1810	5
2	Packets Per Second	0.4	3.7	0
3	Packets Per Minute	24.2	223.4	2.7
4	Packets Per Hour	1451.3	13404	162
5	Request Packets	[235,1328]	1791	3
6	Requests Per Second	0.2	3.7	0
7	Requests Per Minute	12.1	221	1.6
8	Requests Per Hour	723.9	13260	96
9	Response Packets	[115,1135]	19	2
10	Responses Per Second	0.2	0	0
11	Responses Per Minute	12.1	2.3	1.1
12	Responses Per Hour	726.8	138	66
13	Source Addresses	7	6	2
14	Destination Addresses	8	204	2
15	Request Source Addresses	5	6	1
16	Request Destination Addresses	8	204	1
17	Response Source Addresses	6	5	1
18	Response Destination Addresses	4	6	1
ID	Layer: **Inernet** Protocol: **ICMP** Scope: **Global Network-level**		Example Sessions	
	Feature Name	Feature Value	Session 'A'	Session 'B'
1	Packet Count	[0,425]	19	1500
2	Packets Per Second	0	0	13.6
3	Packets Per Minute	3.2	2.3	815.1
4	Packets Per Hour	189.6	138	48906
5	Error Packets	[0,417]	0	198
6	Error Per Second	0	0	1.8
7	Error Per Minute	3.1	0	107.6
8	Error Per Hour	188	0	6456
9	Query Packets	[0,8]	19	1302
10	Query Packets Per Second	0	0	11.8
11	Query Packets Per Minute	0	2.3	707.5
12	Query Packets Per Hour	1.6	138	42450
13	Query Request Packets	[0,4]	9	786
14	Query Request Packets Per Second	0	0	7.1
15	Query Request Packets Per Minute	0	1.1	427.1
16	Query Request Packets Per Hour	0.8	66	25626
17	Query Response Packets	[0,4]	10	516
18	Query Response Packets Per Second	0	0	4.7
19	Query Response Packets Per Minute	0	1.2	280.4
20	Query Response Packets Per Hour	0.8	72	16824
21	Source Addresses	2	5	2
22	Destination Addresses	2	6	2
23	Request Source Addresses	1	1	1
24	Request Destination Addresses	1	5	1
25	Response Source Addresses	1	4	1
26	Response Destination Addresses	1	1	1
27	Maximum Packet Size	84	60	56
28	Connection Count	[0,2]	5	187
29	Connections Per Second	0	0	1.7
30	Connections Per Minute	0	0.6	101.6
31	Connections Per Hour	0.2	36	6096

Table A.11: ARP and ICMP Global Network-level features

It is clear from the table and the highlighted cells, that we perform comparison of session features values against their corresponding thresholds of

normal behavior within the network anomaly profile.

The Greyed cells indicate the violated thresholds for respective features. The comparison process indicates the findings in Table A.12 below.

Packet Capture Session	Protocol	Layer
Session 'A'	ARP ICMP	Network Interface Internet
Session 'B'	ICMP	Internet

Table A.12: Network Profile Results

The Results from utilizing the Network Anomaly Profile, compared to originally existing attacks within the sessions, proves the efficiency of utilizing the profile in Anomaly ID.

Appendix B

Verification of logical connections for UDP-based application protocols

In this appendix we will demonstrate the validity of our proposed methodology of identifying and reconstructing logical UDP-based connections. The demonstration will include the experimental process and the associated data analysis of a series of experiments. UDP and its associated application protocols are used in real life in one of two main communication situations: unicast and multicast communication. We will represent UDP-based applications within these networking scenarios:

1. Unicast communication
2. Multicast communication
3. Intrusion Detection

In unicast communication, there is only one source and one destination. The source and destination addresses in the IP packet are the unicast addresses. In multicast communication, there is only one source and a group of destinations identified by a multicast address. The source and destination addresses in the IP packet are the source address and the group address respectively [1].

As real life examples, UDP-based unicast communication can be examined using any UDP-based application protocol whose behavior and specifications require unicast communication. Also, UDP-based multicast communication can be investigated using any UDP-based media streaming protocol.

To demonstrate the validity of our proposed methodology in unicast communication we are going to experimentally examine and analyze DNS protocol, in multicast communication we are going to experimentally investigate and analyze RTP [75][76][77], and finally for Intrusion detection we utilize NBNS and LLMNR protocol.

B.1 Unicast Communication

In order to investigate unicast communication scenarios we captured real world traffic of a computer connected to the Internet from campus network of Electrical and Computer Engineering Department, University of Alberta in Edmonton. This situation guarantees the preciseness of our proposed methodology via examining a variety of protocols in normal uncontrolled environment. We performed three browsing scenarios to test our methodology, we will begin our explanation of such scenarios from the easiest to the hardest.

B.1.1 Scenario 1

In this scenario we initiated a simple browsing session to browse two websites simultaneously; "*Google*" and "*Facebook*". We used Mozilla Firefox web browser to browse the websites in two separate tabs. We started capturing network traffic, then browsed both websites, closed the tabs, closed the browser, then we terminated the packet capture session. The IP address of the browsing computer

is "*129.128.212.139*".

	Quantitative information			Temporal information		
Protocol	Packets	Bytes (K)	Traffic %	Start time	End time	Duration
TCP	874	248,707	92.69	13:22:32	13:24:45	00:02:12
UDP	153	19,604	7.31			
	1027	268,311	100			

(a) Scenario 1 Summary information

Protocol	Packets	Bytes (K)	UDP %	Selected?
DNS	118	13,111	66.88	Yes
Multicast DNS	2	0,210	1.07	No
Dropbox LanSync Discovery	33	6,283	32.05	No
3 applications	153	19,604	100	1 selected

(b) Scenario 1 UDP Detailed Information

Table B.1: Scenario 1: Packet Capture Information

Table B.1a indicates that we have 874 TCP packets and 153 UDP packets. Table B.1b illustrates the captured UDP-based application protocols, we only process DNS protocol. We identified and reconstructed 36 TCP connections and 59 UDP-based logical connections from DNS packets.

According to the activities that we did in this scenario we browsed "*Google*" and "*Facebook*" websites. Thus, connection-wise investigation should reflect these activities. Within our identified connections we should see TCP-based HTTPS connections to browse these websites. In addition, these browsing connections should be preceded by DNS connections that query the names "*Google*" and "*Facebook*" respectively.

We found these traffic patterns within our reconstructed connections, Table B.2.

ConnectionID	Start Time	Duration	Service	Protocol	Source Port	Source IP	Destination Port	Destination IP	Status	Packets	Data
48	13:22:45.990	00:00:00.160	DNS	UDP	52842	129.128.212.139	53	129.128.208.6	Complete	2	134
8	13:22:46.303	00:01:42.837	HTTPS	TCP	51857	129.128.212.139	443	216.58.193.67	Complete	146	99597
75	13:23:23.027	00:00:00.096	DNS	UDP	61912	129.128.212.139	53	129.128.208.6	Complete	2	169
15	13:23:23.213	00:01:05.770	HTTPS	TCP	51866	129.128.212.139	443	66.220.146.36	Complete	51	27317

Table B.2: Scenario 1: Summary Connections

DNS connection 46 in Table B.2 is a connection issued to the DNS server to

know the IP address of "*Google*" website. Figure B.1 illustrates such connection.

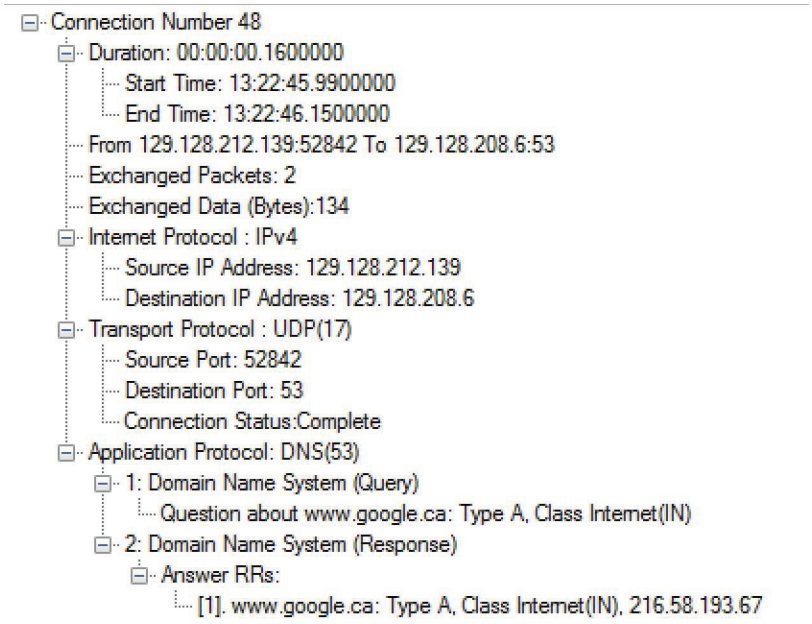


Figure B.1: Google DNS Query Connection

It is clear from Figure B.1 that the DNS connection asks for the IPv4 address of "*Google*" website, and the DNS server responded that it is *216.58.193.67*. Then from Table B.2, connection 8, the server started browsing *Google* using HTTPS protocol over TCP.

Likewise, DNS connection 75 in Table B.2 is a connection to DNS server to find out the IP address of "*Facebook*" website. Figure B.2 shows the details of such DNS connection.

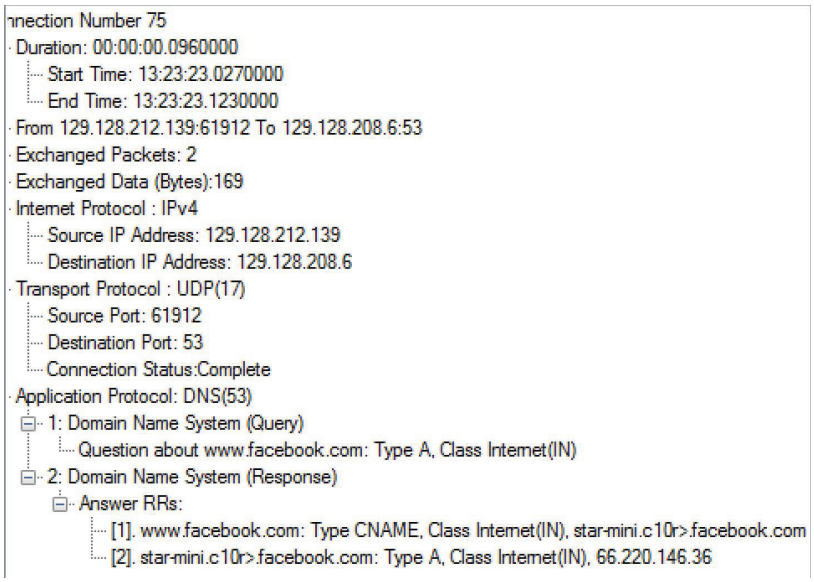


Figure B.2: Facebook DNS Query Connection

Again, it is obvious that the IPv4 address of "*Facebook*" is *66.220.146.36*. The computer uses that address in the next connection to connect to *Facebook*.

B.1.2 Scenario 2

In this experiment we needed to make it a little bit harder for our algorithm to identify and reconstruct logical connections by trying to invoke the first shortcoming that the transport address alone is not sufficient. In other words, we needed the computer to initiate multiple DNS connections using the same source port.

So, instead of having a simple browsing session containing just two different websites, we opened and closed multiple websites (around *50* websites) in the same packet capture session using the same browser (Mozilla Firefox). Additionally, we decreased Windows available UDP dynamic port range to its minimum (*255* ports), to force the computer to use the same source port number. The IP address of the browsing computer is "*129.128.212.139*". Table B.3

illustrates information about the captured traffic within this scenario.

	Quantitative information			Temporal information		
Protocol	Packets	Bytes (K)	Traffic %	Start time	End time	Duration
TCP	3129	940,208	94.5	12:13:05	12:21:12	00:08:06
UDP	461	54,392	5.5			
	3590	994,600	100			

(a) Scenario 2 Summary information

Protocol	Port	Packets	Bytes (K)	UDP %	Selected?
DNS	53	205	23,353	42.94	Yes
NBNS	137	60	5,040	9.27	Yes
NBDS	138	27	5,848	10.75	Yes
LLMNR	5355	74	3,784	6.95	Yes
SSDP	1900	29	4,466	8.21	Yes
BOOTP	67,68	2	0,663	1.22	Yes
Dropbox	17500	62	10,774	19.8	No
McAfee	6646	2	0,464	0.86	No
8 applications		461	54,392	100	6 selected

(b) Scenario 2 UDP Detailed Information

Table B.3: Scenario 2: Packet Capture Information

By applying our algorithm to this capture traffic, our algorithm was able to identify and reconstruct 345 connections. We processed all TCP packets (*3129* packets) and identified 117 TCP connections. Also, we processed 397 UDP packets and skipped 64 packets (*2* packets for *McAfee*, *62* packets for *Dropbox LanSync Discovery*) and identified 228 UDP logical connections. Table B.4 illustrates the identified and reconstructed 228 UDP logical connections with their respective statuses.

Application	Connections	Status	
		Complete	Incomplete
DNS	106	99	7
NBNS	19	0	19
NBDS	27	0	27
LLMNR	45	0	45
SSDP	29	0	29
BOOTP	2	0	2
Total	228	99	129

Table B.4: Scenario 2: UDP Connections

We achieved the goal from our experiment, the target computer re-used the same dynamic port number to initiate several UDP logical connections. The

packets in Table B.5 shows such situation.

PacketID	Timestamp	Source Port	Source IP	Destination Port	Destination IP	Data
3066	12:17:20.263	50061	129.128.212.139	53	129.128.208.6	61
3067	12:17:20.390	53	129.128.208.6	50061	129.128.212.139	89
3903	12:19:20.750	50061	129.128.212.139	53	129.128.208.6	72
3909	12:19:21.740	53	129.128.208.6	50061	129.128.212.139	427
4107	12:19:51.510	50061	129.128.212.139	53	129.128.208.6	72
4108	12:19:51.637	53	129.128.208.6	50061	129.128.212.139	88

Table B.5: Scenario 2: Re-occurring Transport Address Packets (Internet and Transport Layer Data)

All these packets, Table B.5, share the same transport address. They have the same port numbers and they are exchanged between the same hosts using UDP. Researchers might try to group these packets using their timestamps but with no justifiable evidence. According to our proposed algorithm, when we add the DNS ID field to the transport address of these we achieve unique distinction between these packets into logical connections. Additionally, the DNS Name of these packets supports our claim, Table B.6.

PacketID	Timestamp	Source Port	Source IP	Destination Port	Destination IP	Data	ID	DNS Name
3066	12:17:20.263	50061	129.128.212.139	53	129.128.208.6	61	10425	plus.google.com
3067	12:17:20.390	53	129.128.208.6	50061	129.128.212.139	89	10425	plus.google.com
3903	12:19:20.750	50061	129.128.212.139	53	129.128.208.6	72	21310	tiles.services.mozilla.com
3909	12:19:21.740	53	129.128.208.6	50061	129.128.212.139	427	21310	tiles.services.mozilla.com
4107	12:19:51.510	50061	129.128.212.139	53	129.128.208.6	72	46336	www.beartracks.ualberta.ca
4108	12:19:51.637	53	129.128.208.6	50061	129.128.212.139	88	46336	www.beartracks.ualberta.ca

Table B.6: Scenario 2: Re-occurring Transport Address Packets (Application Layer Data)

Now that we achieved DNS logical connections we can reconstruct their related fields. Table B.7 illustrates our identified and reconstructed DNS logical connections.

ConnectionID	Start Time	Duration	Service	Protocol	Source Port	Source IP	Destination Port	Destination IP	Status	Packets	Data
170	12:17:20.263	00:00:00.127	DNS	UDP	50061	129.128.212.139	53	129.128.208.6	Complete	2	150
185	12:19:20.750	00:00:00.990	DNS	UDP	50061	129.128.212.139	53	129.128.208.6	Complete	2	499
202	12:19:51.510	00:00:00.127	DNS	UDP	50061	129.128.212.139	53	129.128.208.6	Complete	2	160

Table B.7: Scenario 2: Re-occurring Transport Address Packets (Reconstructed Connections)

B.1.3 Scenario 3

This experiment is similar to scenario 2, the difference is that browsing was done using 3 browsers simultaneously to intensify the produced network traffic. Again, We set Microsoft Windows available UDP dynamic port range to its minimum (*255* ports), to force the computer to use the same source port number with multiple UDP connections. The IP address of the browsing computer is "*129.128.212.139*".

	Quantitative information			Temporal information		
Protocol	Packets	Bytes (K)	Traffic %	Start time	End time	Duration
TCP	3532	1,026,657	90.21	16:31:39	16:42:25	00:10:45
UDP	383	45,895	9.79			
	3915	1,072,552	100			

(a) Scenario 3 Summary information

Protocol	Port	Packets	Bytes (K)	UDP %	Selected?
Dropbox	17500	53	10,441	13.83	No
DNS	53	291	30,227	76	Yes
BOOTP	67,68	4	1,312	1.05	Yes
LLMNR	5355	10	500	2.61	Yes
Multicast DNS	5353	5	626	1.31	No
SSDP	1900	12	1,845	3.13	Yes
NBNS	137	6	468	1.57	Yes
NBDS	138	2	476	0.5	Yes
8 applications		383	45,895	100	6 selected

(b) Scenario 3 UDP Detailed Information

Table B.8: Scenario 3: Packet Capture Information

By applying our algorithm to this captured traffic, our algorithm was able to identify 324 overall connections; 147 TCP connections and 177 UDP logical connections. The 147 TCP were identified and reconstructed out of 3532 TCP packets. The 177 UDP logical connections were identified and reconstructed by processing 325 uDP packets and skipping 58 packets (5 packets for Multicast DNS, 53 packets for Dropbox LanSync Discovery), tbs3Info.

Application	Connections	Status	
		Complete	Incomplete
DNS	155	136	19
NBNS	2	0	2
NBDS	1	0	1
LLMNR	3	0	3
SSDP	12	0	12
BOOTP	4	0	4
Total	177	136	41

Table B.9: Scenario 3: UDP Connections

Table B.9 illustrates the reconstructed 177 UDP logical connections detailed by application and connection status. Our activities within this capture session to achieve the shortcoming of re-occuring transport address were successful,Table B.10.

PacketID	Timestamp	Source Port	Source IP	Destination Port	Destination IP	Data
4869	16:32:23.157	50246	129.128.212.139	53	129.128.208.6	69
4870	16:32:23.247	53	129.128.208.6	50246	129.128.212.139	101
5292	16:33:13.640	50246	129.128.212.139	53	129.128.208.6	64
5296	16:33:14.093	53	129.128.208.6	50246	129.128.212.139	242
7503	16:38:51.467	50246	129.128.212.139	53	129.128.208.6	69
7504	16:38:51.650	53	129.128.208.6	50246	129.128.212.139	101
6054	16:35:07.737	50249	129.128.212.139	53	129.128.208.6	58
6057	16:35:08.113	53	129.128.208.6	50249	129.128.212.139	97
8140	16:40:44.670	50249	129.128.212.139	53	129.128.208.6	69
8141	16:40:44.827	53	129.128.208.6	50249	129.128.212.139	101
7108	16:37:44.897	50187	129.128.212.139	53	129.128.208.6	69
7109	16:37:45.027	53	129.128.208.6	50187	129.128.212.139	101
7589	16:39:06.847	50187	129.128.212.139	53	129.128.208.6	69
7590	16:39:06.980	53	129.128.208.6	50187	129.128.212.139	141
8138	16:40:44.317	50187	129.128.212.139	53	129.128.208.6	69
8139	16:40:44.503	53	129.128.208.6	50187	129.128.212.139	101

Table B.10: Scenario 3: Re-occurring Transport Address Packets (Internet and Transport Layer Data)

We ordered the packets in Table B.10 based on their hosts and port numbers. It is clear that there 3 groups of packets that share the same transport address. Again, there is no definite evidence that says they are 3 distinct groups. Researchers may try to associate the DNS name with these packets to form unique groups, Table B.11.

Grouping	PacketID	Timestamp	Source Port	Source IP	Destination Port	Destination IP	Data	DNS Name
Group 1	4869	16:32:23.157	50246	129.128.212.139	53	129.128.208.6	69	syndication.twitter.com
	4870	16:32:23.247	53	129.128.208.6	50246	129.128.212.139	101	syndication.twitter.com
	5292	16:33:13.640	50246	129.128.212.139	53	129.128.208.6	64	checkip.dyndns.org
	5296	16:33:14.093	53	129.128.208.6	50246	129.128.212.139	242	checkip.dyndns.org
	7503	16:38:51.467	50246	129.128.212.139	53	129.128.208.6	69	syndication.twitter.com
	7504	16:38:51.650	53	129.128.208.6	50246	129.128.212.139	101	syndication.twitter.com
Group 2	6054	16:35:07.737	50249	129.128.212.139	53	129.128.208.6	58	www.bing.com
	6057	16:35:08.113	53	129.128.208.6	50249	129.128.212.139	97	www.bing.com
	8140	16:40:44.670	50249	129.128.212.139	53	129.128.208.6	69	syndication.twitter.com
	8141	16:40:44.827	53	129.128.208.6	50249	129.128.212.139	101	syndication.twitter.com
Group 3	7108	16:37:44.897	50187	129.128.212.139	53	129.128.208.6	69	syndication.twitter.com
	7109	16:37:45.027	53	129.128.208.6	50187	129.128.212.139	101	syndication.twitter.com
	7589	16:39:06.847	50187	129.128.212.139	53	129.128.208.6	69	syndication.twitter.com
	7590	16:39:06.980	53	129.128.208.6	50187	129.128.212.139	141	syndication.twitter.com
	8138	16:40:44.317	50187	129.128.212.139	53	129.128.208.6	69	syndication.twitter.com
	8139	16:40:44.503	53	129.128.208.6	50187	129.128.212.139	101	syndication.twitter.com

Table B.11: Scenario 3: Re-occurring Transport Address Packets (Internet and Transport Layer Data + DNS Name)

Table B.11 illustrates the packet grouped by their transport address. Researchers might try to provide distinctive groups by attaching the DNS Name to the transport address. In this case, they might choose to regroup the packets of **'Group 1'** into 2 or 3 groups, and re-group the packets of **'Group 2'** into 2 groups, and no extra re-grouping can be made to packets of **'Group 3'** since they all share the same DNS name and transport address. Our algorithm analyzes these packets based on DNS ID field to produce Table B.12.

Grouping	PacketID	Timestamp	Source Port	Source IP	Destination Port	Destination IP	Data	DNS Name	ID
Group 1	4869	16:32:23.157	50246	129.128.212.139	53	129.128.208.6	69	syndication.twitter.com	13812
	4870	16:32:23.247	53	129.128.208.6	50246	129.128.212.139	101	syndication.twitter.com	13812
	5292	16:33:13.640	50246	129.128.212.139	53	129.128.208.6	64	checkip.dyndns.org	1195
	5296	16:33:14.093	53	129.128.208.6	50246	129.128.212.139	242	checkip.dyndns.org	1195
	7503	16:38:51.467	50246	129.128.212.139	53	129.128.208.6	69	syndication.twitter.com	30224
	7504	16:38:51.650	53	129.128.208.6	50246	129.128.212.139	101	syndication.twitter.com	30224
Group 2	6054	16:35:07.737	50249	129.128.212.139	53	129.128.208.6	58	www.bing.com	60318
	6057	16:35:08.113	53	129.128.208.6	50249	129.128.212.139	97	www.bing.com	60318
	8140	16:40:44.670	50249	129.128.212.139	53	129.128.208.6	69	syndication.twitter.com	36484
	8141	16:40:44.827	53	129.128.208.6	50249	129.128.212.139	101	syndication.twitter.com	36484
Group 3	7108	16:37:44.897	50187	129.128.212.139	53	129.128.208.6	69	syndication.twitter.com	57841
	7109	16:37:45.027	53	129.128.208.6	50187	129.128.212.139	101	syndication.twitter.com	57841
	7589	16:39:06.847	50187	129.128.212.139	53	129.128.208.6	69	syndication.twitter.com	11900
	7590	16:39:06.980	53	129.128.208.6	50187	129.128.212.139	141	syndication.twitter.com	11900
	8138	16:40:44.317	50187	129.128.212.139	53	129.128.208.6	69	syndication.twitter.com	14996
	8139	16:40:44.503	53	129.128.208.6	50187	129.128.212.139	101	syndication.twitter.com	14996

Table B.12: Scenario 3: Re-occurring Transport Address Packets (Internet and Transport Layer Data + DNS Name + ID)

Table B.12 shows the same packets as in Table B.11 but we added DNS ID field to them. Our algorithm groups packets into logical connections by adding the ID field to the transport address. Therefore, the packets of **'Group 1'** are actually 3

distinct logical connections. The packets of **'Group 2'** are 2 logical connections, and the packets in **'Group 3'** are 3 logical connections, not one connection.
The interesting idea about this scenario is that despite the packets share the same transport address and the same DNS name, this does not necessarily indicate that they are one group. Thus, this proves the inefficiency of the DNS name field and contrarily proves the validity of our claim.
Furthermore, even if this analysis of these packets is somehow indicative based on DNS name, this might be only valid for DNS protocol, not for all UDP-based application protocols. Next, we verify our hypothesis for multicast UDP application protocols.

B.2 Multicast Communication

In this section we captured network traffic of an isolated small computer network containing 3 computers. We performed a streaming scenarios to test our algorithm, we will begin our explanation from the easiest to the hardest.

B.2.1 Scenario 4

In this experiment, we used the streaming feature of VLC media player to stream a short video in the isolated network and then we analyzed the traffic. We customized VLC to send the video using RTP. The IP address of the streaming computer is *"192.168.10.200"*. Table B.13 shows detailed information about the traffic captured during this scenario.

	Quantitative information			Temporal information		
Protocol	Packets	Bytes (K)	Traffic %	Start time	End time	Duration
IGMP	3	120	0.26	14:21:17	14:23:44	00:02:26
UDP	1146	1,440,117	99.74			
	1149	1,440,237	100			

(a) Scenario 4 Summary information

Protocol	Port	Packets	Bytes (K)	UDP %	Selected?
LLMNR	5355	8	400	0.7	Yes
NBNS	137	20	1560	1.74	Yes
SSDP	1900	39	5695	3.4	Yes
NBDS	138	5	1145	0.43	Yes
RTP	5004	1054	1429224	92	Yes
RTCP	5005	19	1748	1.65	Yes
Unofficial	62976	1	345	0.08	No
7 applications		1146	1,440,237	100	6 selected

(b) Scenario 4 UDP Detailed Information

Table B.13: Scenario 4: Packet Capture Information

By applying our algorithm to this capture session, the algorithm skipped 1 packet (unofficial packet) and processed the rest. The algorithm was able to identify and reconstruct 26 UDP connections out of 1145 packets. Our algorithm was able to reconstruct 3 LLMNR, 3 NBDS, 3 NBNS, 15 SSDP, and 1 RTP/RTCP logical connections. Table B.13 represent summary and detailed information for scenario 4.

According to the specifications in [75][76] RTP works in tandem with RTCP, RTP uses a transmission channel over port 5004 to send the actual file to be transmitted. RTCP aids RTP during the transmission of the file, it uses control channel over port 5005. The control channel is used to monitor the quality of transmission by sending information to participating hosts within the transmission.

Looking at these 2 protocols from the perspective of logical connections, since it is one transmitted file and its companion control information, the packets transmitted by these protocols can be regarded as one logical connection. We process their respective logical connection by grouping their packets with the

32 bit synchronization source identifier (SSRC) field, and we reconstructed the following connection, Table B.14.

Start Time	End Time	Duration	Protocol	Source Port	Source IP	Destination Port	Destination IP	Status	Packets	Data
14:21:39.477	14:21:39.367	00:00:00.110	RTP(5004)	1038	192.168.10.200	5004	239.0.0.1	Incomplete	1,054	1,429,224
14:21:40.837	14:21:30.633	00:00:10.204	RTCP(5005)	1039	192.168.10.200	5005	239.0.0.1	Incomplete	19	1,748

Table B.14: Scenario 4: Identification and Reconstruction of Stream Connections

Table B.14 depicts the success of our algorithm in identifying and reconstructing the stream logical connection, though it takes place using two different transport addresses via ports 5004 and 5005.

B.3 Intrusion Detection

In this section we want to show how our proposed algorithm can be useful in Intrusion Detection. We generate a networking scenario to show that our algorithm is useful in easing the process of Intrusion Detection. The scenario includes the traffic of InsideSniffer attack. We execute the attack in its elusive pattern using LLMNR protocols.

B.3.1 Scenario 5

In this scenario we executed InsideSniffer attack using LLMNR protocol. The attacker computer is *'192.168.10.5'* and the attack targets the range of 10 IP addresses from *'192.168.10.1'* to *'192.168.10.10'*. Table B.15 illustrates summary and detailed information about the captured packets within the scenario.

	Quantitative information			Temporal information		
Protocol	Packets	Bytes (K)	Traffic %	Start time	End time	Duration
UDP	86	8,979	94	17:29:43	17:30:16	00:00:33
ICMP	4	582	6			
	90	9,561	100			

(a) Scenario 5 Summary information

Protocol	Port	Packets	Bytes (K)	UDP %	Selected?
LLMNR	5355	46	3,362	9.27	Yes
NBNS	137	7	832	53.29	Yes
SSDP	1900	33	4,785	37.44	Yes
3 applications		86	8,979	100	3 selected

(b) Scenario 5 UDP Detailed Information

Table B.15: Scenario 5: Packet Capture Information

This packet capture session is small and is intended to capture the attack pattern only. Table B.16 and Table B.17 show the packets that form the attack pattern. Table B.16 illustrates their Internet and Transport Layer information.

PacketID	Timestamp	Source Port	Source IP	Destination Port	Destination IP	Data
16937	17:29:49.123	53252	192.168.10.5	5355	224.0.0.252	71
16938	17:29:49.317	53252	192.168.10.5	5355	224.0.0.252	71
16939	17:29:49.483	53252	192.168.10.5	5355	224.0.0.252	71
16940	17:29:49.637	59008	192.168.10.5	5355	224.0.0.252	71
16941	17:29:49.777	59008	192.168.10.5	5355	224.0.0.252	71
16942	17:29:49.930	59008	192.168.10.5	5355	224.0.0.252	71
16943	17:29:50.083	5355	192.168.10.2	59008	192.168.10.5	116
16944	17:29:50.250	5355	192.168.10.2	59008	192.168.10.5	116
16945	17:29:50.410	5355	192.168.10.2	59008	192.168.10.5	116
16954	17:29:53.237	55223	192.168.10.5	5355	224.0.0.252	71
16955	17:29:53.413	55223	192.168.10.5	5355	224.0.0.252	71
16956	17:29:53.580	55223	192.168.10.5	5355	224.0.0.252	71
16957	17:29:53.743	5355	192.168.10.3	55223	192.168.10.5	119
16958	17:29:53.910	5355	192.168.10.3	55223	192.168.10.5	119
16966	17:29:55.290	50250	192.168.10.5	5355	224.0.0.252	71
16967	17:29:55.480	50250	192.168.10.5	5355	224.0.0.252	71
16968	17:29:55.663	50250	192.168.10.5	5355	224.0.0.252	71
16969	17:29:55.850	63428	192.168.10.5	5355	224.0.0.252	71
16970	17:29:56.040	63428	192.168.10.5	5355	224.0.0.252	71
16971	17:29:56.240	63428	192.168.10.5	5355	224.0.0.252	71
16972	17:29:56.427	50437	192.168.10.5	5355	224.0.0.252	71
16973	17:29:56.627	50437	192.168.10.5	5355	224.0.0.252	71
16974	17:29:56.837	50437	192.168.10.5	5355	224.0.0.252	71
16975	17:29:57.667	58874	192.168.10.5	5355	224.0.0.252	71
16976	17:29:57.880	58874	192.168.10.5	5355	224.0.0.252	71
16977	17:29:58.083	58874	192.168.10.5	5355	224.0.0.252	71
16978	17:29:58.293	56012	192.168.10.5	5355	224.0.0.252	71
16979	17:29:58.493	56012	192.168.10.5	5355	224.0.0.252	71
16980	17:29:58.803	56012	192.168.10.5	5355	224.0.0.252	71
16981	17:29:59.117	63897	192.168.10.5	5355	224.0.0.252	72
16982	17:29:59.363	63897	192.168.10.5	5355	224.0.0.252	72
16983	17:29:59.593	63897	192.168.10.5	5355	224.0.0.252	72

Table B.16: Scenario 5 Attack Packets: Internet and Transport Layer Data

Table B.17 shows the Application Header data of the packets in Table B.16.

PacketID	QR	ID	QName	QType	QClass	RData
16937	0	45981	1.10.168.192.in-addr.arpa	12	1	–
16938	0	45981	1.10.168.192.in-addr.arpa	12	1	–
16939	0	45981	1.10.168.192.in-addr.arpa	12	1	–
16940	0	5344	2.10.168.192.in-addr.arpa	12	1	–
16941	0	5344	2.10.168.192.in-addr.arpa	12	1	–
16942	0	5344	2.10.168.192.in-addr.arpa	12	1	–
16943	1	5344	2.10.168.192.in-addr.arpa	12	1	syspam
16944	1	5344	2.10.168.192.in-addr.arpa	12	1	syspam
16945	1	5344	2.10.168.192.in-addr.arpa	12	1	syspam
16954	0	39884	3.10.168.192.in-addr.arpa	12	1	–
16955	0	39884	3.10.168.192.in-addr.arpa	12	1	–
16956	0	39884	3.10.168.192.in-addr.arpa	12	1	–
16957	1	39884	3.10.168.192.in-addr.arpa	12	1	syspah-pc
16958	1	39884	3.10.168.192.in-addr.arpa	12	1	syspah-pc
16966	0	42674	4.10.168.192.in-addr.arpa	12	1	–
16967	0	42674	4.10.168.192.in-addr.arpa	12	1	–
16968	0	42674	4.10.168.192.in-addr.arpa	12	1	–
16969	0	22650	6.10.168.192.in-addr.arpa	12	1	–
16970	0	22650	6.10.168.192.in-addr.arpa	12	1	–
16971	0	22650	6.10.168.192.in-addr.arpa	12	1	–
16972	0	59798	7.10.168.192.in-addr.arpa	12	1	–
16973	0	59798	7.10.168.192.in-addr.arpa	12	1	–
16974	0	59798	7.10.168.192.in-addr.arpa	12	1	–
16975	0	12182	8.10.168.192.in-addr.arpa	12	1	–
16976	0	12182	8.10.168.192.in-addr.arpa	12	1	–
16977	0	12182	8.10.168.192.in-addr.arpa	12	1	–
16978	0	42816	9.10.168.192.in-addr.arpa	12	1	–
16979	0	42816	9.10.168.192.in-addr.arpa	12	1	–
16980	0	42816	9.10.168.192.in-addr.arpa	12	1	–
16981	0	37091	10.10.168.192.in-addr.arpa	12	1	–
16982	0	37091	10.10.168.192.in-addr.arpa	12	1	–
16983	0	37091	10.10.168.192.in-addr.arpa	12	1	–

Table B.17: Scenario 5 Attack Packets: Application Layer Data

Tables B.16 and B.17 show the InsideSniffer attack packets. By examining *'PacketID'*, *'Source IP'*, *'Source Port'*, *'Destination IP'*, and *'Destination Port'* fields in Table B.16, and '*PacketID*', '*QR*', '*ID*', '*QName*', and '*RData*' fields in Table B.17, it becomes clear that the attacker *192.168.10.5* scanned the IP address range from *192.168.10.1* to *192.168.10.10*.

It is obvious from Table B.17, LLMNR QR field, the LLMNR request (0) and response (1) packets, the packets signify that the attacker was successfully able to resolve IP addresses *2.10.168.192.in-addr.arpa* and *3.10.168.192.in-addr.arpa* to their respective names, since we have response packets in the traffic.

Appendix C

Features of Multi-Perspective Network Traffic Description

C.1 Network Interface Layer Features

ID	Layer: **Network Interface**			
	Protocol	Scope	Auxiliary Perspectives	Feature Name
1	ARP	Network	Packet Temporal	Packet Count
2				Packets Per Second
3				Packets Per Minute
4				Packets Per Hour
5				Request Packets
6				Requests Per Second
7				Requests Per Minute
8				Requests Per Hour
9				Response Packets
10				Responses Per Second
11				Responses Per Minute
12				Responses Per Hour
13			Address	Source Addresses
14				Destination Addresses
15				Request Source Addresses
16				Request Destination Addresses
17				Response Source Addresses
18				Response Destination Addresses
19		Host	Address	Destinations Per Source
20			Packet	Requests Per Destinations
21				Overall Requests Per Source
22				Responses Per Sources
23				Overall Responses Per Destination

Table C.1: Network Interface Layer Features

C.2 Internet Layer Features

ID	Layer: **Internet**			
	Protocol	Scope	Auxiliary Perspectives	Feature Name
1	ICMP	Network	Packet Temporal	Packet Count
2				Packets Per Second
3				Packets Per Minute
4				Packets Per Hour
5				Error Packets
6				Error Per Second
7				Error Per Minute
8				Error Per Hour
9				Query Packets
10				Query Packets Per Second
11				Query Packets Per Minute
12				Query Packets Per Hour
13				Query Request Packets
14				Query Request Packets Per Second
15				Query Request Packets Per Minute
16				Query Request Packets Per Hour
17				Query Response Packets
18				Query Response Packets Per Second
19				Query Response Packets Per Minute
20				Query Response Packets Per Hour
21			Address	Source Addresses
22				Destination Addresses
23				Request Source Addresses
24				Request Destination Addresses
25				Response Source Addresses
26				Response Destination Addresses
27			Packet	Maximum Packet Size
28			Connection	Connection Count
29			Connection Temporal	Connections Per Second
30				Connections Per Minute
31				Connections Per Hour
32		Host	Address	Destinations Per Source
33			Packet	Requests Per Destinations
34				Overall Requests Per Source
35				Responses Per Sources
36				Overall Responses Per Destination
37			Connection	Connections Per Destination
38				Overall Connections Per Source
39	IGMP	Network	Packet Temporal	Packet Count
40				Packets Per Second
41				Packets Per Minute
42				Packets Per Hour
43				Multicast Packets
44				Multicast Packets Per Second
45				Multicast Packets Per Minute
46				Multicast Packets Per Hour
47				Unicast Packets
48				Unicast Packets Per Second
49				Unicast Packets Per Minute
50				Unicast Packets Per Hour
51		Host		Multicast Packets Per Second
52				Unicast Packets Per Second
53	General	Network	Packet Address	LAND Indicator

Table C.2: Internet Layer Features

C.3 Transport Layer Features

ID	Layer: **Transport**			
	Protocol	Scope	Auxiliary Perspectives	Feature Name
1	TCP	Network	Packet Temporal	Packet Count
2				Packets Per Second
3				Packets Per Minute
4				Packets Per Hour
5			Connection	Connection Count
6			Connection Temporal	Connections Per Second
7				Connections Per Minute
8				Connections Per Hour
9			Protocol	Invoked Services
10			Protocol,Packet	Service Packets
11		Host	Protocol	Invoked Services
12			Packet	Exchanged Packets
13			Connection	Established Connections
14	UDP	Network	Packet Temporal	Packet Count
15				Packets Per Second
16				Packets Per Minute
17				Packets Per Hour
18			Connection	Connection Count
19			Connection Temporal	Connections Per Second
20				Connections Per Minute
21				Connections Per Hour
22			Protocol	Invoked Services
23			Protocol,Packet	Service Packets
24		Host	Protocol	Invoked Services
25			Packet	Exchanged Packets
26			Connection	Established Connections
27	General	Host	Protocol	Invoked Transport Services
28			Packet,Protocol	Exchanged Transport Packets
29			Packet,Connection	Established Transport Connections
30			Protocol	Attempted Transport Services
31			Packet,Protocol	Exchanged Attempted Transport Packets
32			Packet,Connection	Established Attempted Transport Connections

Table C.3: Transport Layer Features

C.4 Application Layer Features

ID	Layer: **Application**			
	Protocol	Scope	Auxiliary Perspectives	Feature Name
1	DNS	Network	Packet Temporal	Packet Count
2				Packets Per Second
3				Packets Per Minute
4				Packets Per Hour
5			Address	MultiCast Destinations
6				Unicast Destinations
7		Host	Protocol	Name Resolution Indicator
8			Address	Destinations Per Source
9	NBNS	Network	Packet Temporal	Packet Count
10				Packets Per Second
11				Packets Per Minute
12				Packets Per Hour
13			Address	MultiCast Destinations
14				Unicast Destinations
15		Host	Protocol	Name Resolution Indicator
16			Address	Destinations Per Source
17	LLMNR	Network	Packet Temporal	Packet Count
18				Packets Per Second
19				Packets Per Minute
20				Packets Per Hour
21			Address	MultiCast Destinations
22				Unicast Destinations
23		Host	Protocol	Name Resolution Indicator
24			Address	Destinations Per Source

Table C.4: Application Layer Features

Appendix D

TCP/IP Protocols

In this Appendix we provide theoretical details about the protocols that we decipher in NetDataCoP.

D.1 Ethernet II Frame (Ethernet Version 2)

Mainly, An Ethernet Frame consists of three main parts; Ethernet header, data, and CRC Checksum. Figure D.1 illustrates the default Ethernet II frame format.

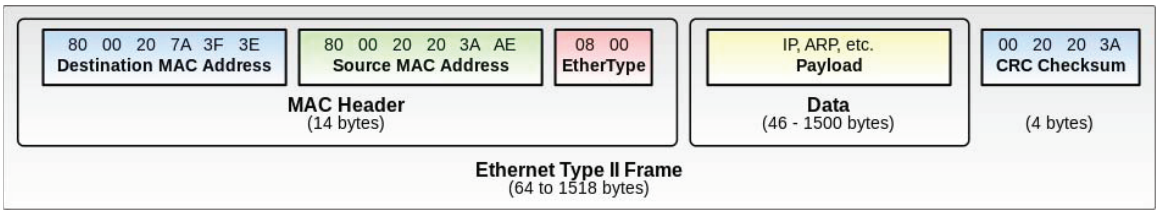


Figure D.1: Ethernet II Frame [3]

The Ethernet frame length ranges from a minimum of 64 bytes to a maximum of 1518 bytes and its fields are:

Destination MAC Address is a 6 byte field that contains the physical address of the destination machine (unicast) or destination machines (multicast or broadcast).

Source MAC Address Likewise, is a 6 byte field that holds the physical address of the source machine that sent the packet.

EtherType is a 2 byte field that determines the type of protocol encapsulated within frame data, *e.g.*, ARP, IP, *etc.* An EtherType of value 0x0800 indicates an IPv4 datagram within frame data (Section 5.2.3), and a value of 0x0806 signifies an ARP frame (next section).

Data is the frame data that ranges from 46 to 1500 bytes, which carries the payload of encapsulated data from upper layer protocols.

CRC Checksum is a 4 byte field used for error detection purposes.

D.2 Address Resolution Protocol (ARP)

As mentioned, ARP is used for logical to physical address mapping. During the address mapping process (Figure D.2), the sender machine sends a broadcast ARP request that holds 4 pieces of information:

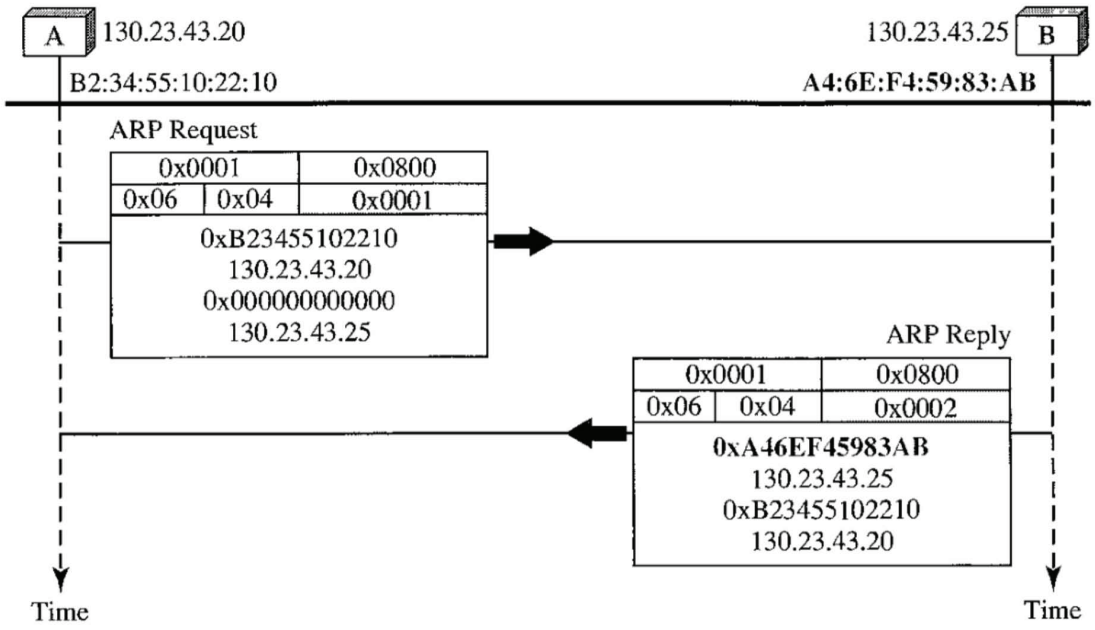


Figure D.2: ARP operation: Logical to Physical Address Mapping Example. [1]

1. *Operation*, Indicates the type of the packet, *i.e.*, Request
2. *Sender Hardware Address*, The physical address of the sender issuing the ARP request
3. *Sender Protocol Address*, The logical address of the sender issuing the ARP request
4. *Target Protocol Address*, The Logical address of the destination

Every host connected to the network receives the broadcast ARP request, but only the intended host recognizes its logical IP address and responds to the request with an ARP Reply that holds the following information:

1. *Operation*, Indicates the type of the packet, *i.e.*, Reply
2. *Sender Hardware Address*, The physical address of the sender responding with the ARP reply

3. *Sender Protocol Address*, The logical address of the sender responding with the ARP reply

4. *Target Hardware Address*, The physical address of the destination that originally issued the ARP request

5. *Target Protocol Address*, The Logical address of the destination that originally issued the ARP request

Having mentioned the mapping process, we explain ARP packet format, Figure D.3.

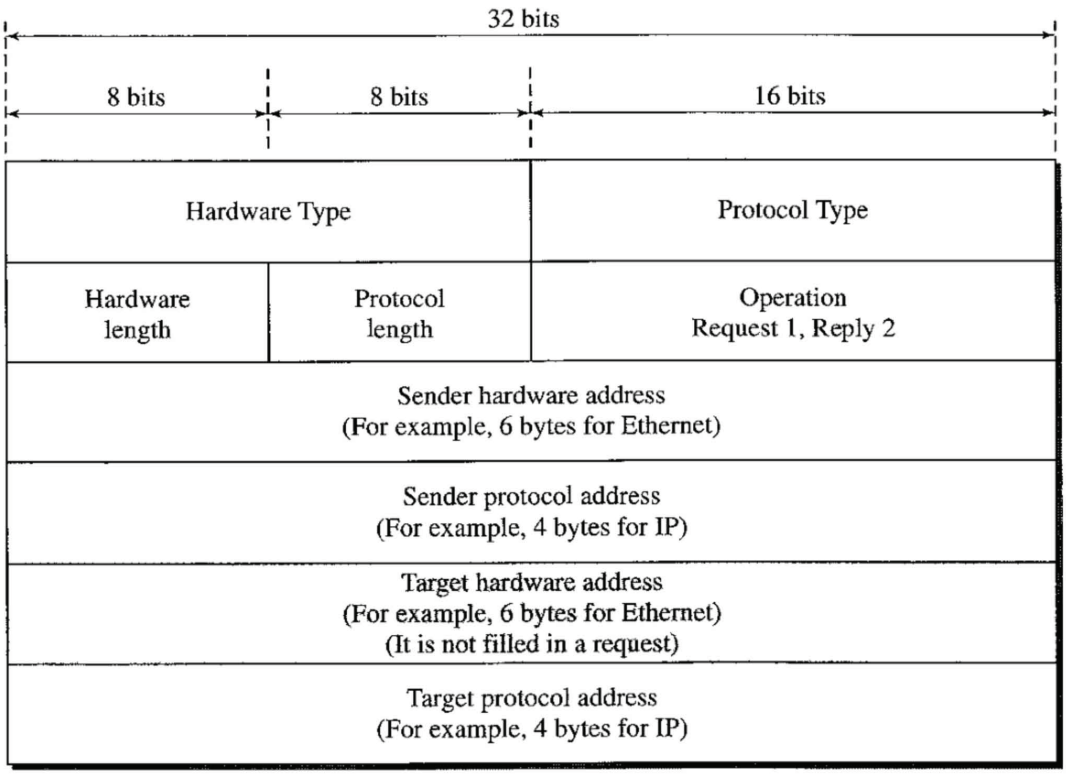


Figure D.3: ARP Packet. [1]

The fields are [1][51]:

Hardware Type is a 16 bit field that defines the type of hardware of the network, a value of 0x0001 indicates Ethernet

Protocol Type is a 16 bit field indicating the protocol, a value of 0x0800 indicates IPv4

Hardware Length is an 8 bit field that defines the length of included physical addresses, a value 6 signifies Ethernet.

Protocol Length is an 8 bit field that defined the length of included logical addresses, a value 4 means IPv4.

Operation is a 16 bit field that indicates the packet type; value 1 for request and 2 for reply.

Sender Hardware Address The physical address of the sender machine whose length is indicated by *Hardware Length* field.

Sender Protocol Address The logical address of the sender machine whose length is indicated by the *Protocol Length* field.

Target Hardware Address The physical address of the destination machine whose length is indicated by *Hardware Length* field.

Target Protocol Address The logical address of the destination machine whose length is indicated by the *Protocol Length* field.

D.3 Internet Protocol version 4 (IPv4)

The Internet Protocol version 4 (IPv4) (Figure D.4) is a best-effort unreliable connectionless datagram protocol utilized by TCP/IP to deliver packets to their intended destinations. We have to distinguish between three different

functionalities needed to deliver packets; naming, addressing, and routing. Naming indicates what we are exploring within a certain host, addressing signifies the location of a host, and routing reveals how to reach that host.
This distinction will eventually lead us to understand the intended functionality of IPv4, among these functionalities IPv4 handles addressing and their related mappings. Lower level protocols deal with routing, and higher level protocols take care of naming. Additionally, IPv4 uses fragmentation to deliver large size packets to their destinations.

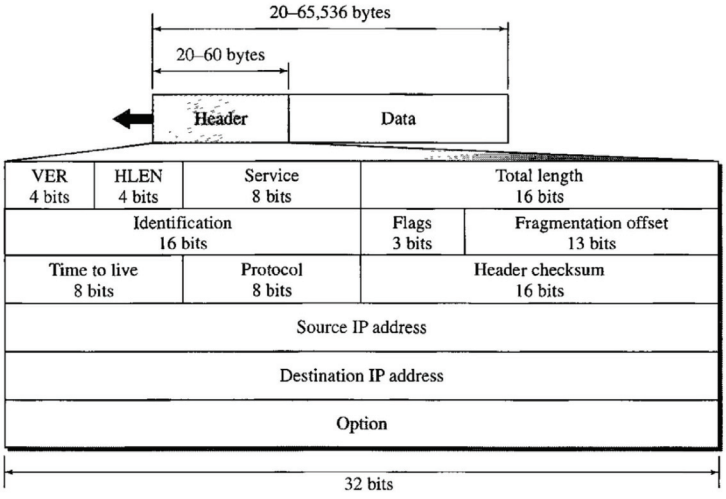


Figure D.4: IPv4 Datagram Format. [1]

Figure D.4 illustrates IPv4 header, its size ranges from 20 to 60 bytes, it contains the necessary information required for packet delivery.

Version It is a 4 bit field that defines the version of the IP datagram. Version 4 indicates an IPv4 datagram, while version 6 signifies an IPv6 datagram.

Header Length (HLEN) A 4 bit field that indicates the length of the header in 32 bit (4 byte) words. A minimum value is 20 bytes (5 fields of 4 bytes = 20 bytes) when no options are specified, otherwise the maximum value is 60

bytes.

Service An 8 bit field, also known as differentiated services or type of service. This field is used to select the optimum service parameters within a particular network.

Total Length It is a 16 bit field that indicates the overall length of the datagram (header + data), *i.e.*, Data Length = Total Length - Header Length.

Identification It is a 16 bit field whose value is assigned by the sender to help the receiver in assembling the datagram fragments.

Flags It is a 3 bit field that indicates if the received datagram is fragmented or not.

Fragmentation Offset If the received datagram is a fragment, this is a 13 bit field that designates the location of this fragment within the datagram.

Time to Live (TTL) It is an 8 bit that defines in seconds the maximum allowed datagram lifetime for it to remain within the network.

Protocol The IPv4 datagram can encapsulate one of many higher level protocols, this is an 8 bit field that indicates the target higher level protocol.

Checksum It is a 16 bit field whose function is to provide error detection for the datagram header.

Source Address It is a 32 bit field that provides the logical address IPv4 address of the source.

Destination Address It is a 32 bit field that holds the logical address IPv4 address of the destination.

D.4 Internet Control Message Protocol (ICMP)

ICMP provides error control and management mechanisms [53][54][55], so ICMP packets has two categories; error-reporting packets and query packets. Table D.1 illustrates the most common ICMP packet types.

ICMP Packet	Sub-Category	Settings	
		Type	Code
Error Reporting	Destination Unreachable	3	0 - Net unreachable
			1 - Host unreachable
			2 - Protocol unreachable
			3 - Port unreachable
			4 - Fragmentation needed
			5 - Source route failed
			6 - Destination network unknown
			7 - Destination host unknown
			8 - Source host isolated
			9 - Communication with destination network is administratively prohibited
			10 - Communication with destination host is administratively prohibited
			11 - Destination network unreachable for TOS
			12 - Destination host unreachable for TOS
	Source Quench	4	0 - none
	Redirection	5	0 - Redirect datagram for the network
			1 - Redirect datagram for the host
			2 - Redirect datagram for the TOS and network
			3 - Redirect datagram for the TOS and host
	Time Exceeded	11	0 - Time to live exceeded in transit
			1 - Fragment reassembly time exceeded
	Parameter Problem	12	0 - Pointer indicates the error
			1 - Missing a required option
			2 - Bad length
Query	Echo Request	8	0
	Echo Reply	0	0
	Router Solicitation	10	0
	Router Advertisement	9	0
	Timestamp Request	13	0
	Timestamp Reply	14	0
	Address Mask Request	17	0
	Address Mask Reply	18	0

Table D.1: ICMP Messages

The explanation of these fields is as follows:

- **Destination Unreachable**, It means that the sent packet can not be

delivered to its intended recipient for one of the reasons indicated by the code field.

- **Source Quench**, It means that the packet has been discarded due to network congestion at some point along the route to the destination.
- **Redirection** This type of packets is intended to update the routing table of network hosts with the IP address of the default router for specific external networks.
- **Time Exceeded** It means that the packet has been discarded because the packet stayed within the network more than allowed, *i.e.*, IPv4 TTL has been exceeded.
- **Parameter Problem** It means that the packet has been discarded due to the presence of ambiguity in one or packet fields.
- **Echo request and reply** This is ping request and reply messages, it is used to check and test if a valid IP level connectivity exists between two particular hosts. The sender host sends a ping request to the receiver host, if a ping reply is sent back to the sender by the receiver this indicates that connectivity between sender and receiver exists.
- **Timestamp request and reply** This type of ICMP packet is used to determine the round-trip time that a datagram takes to move between the sender and receiver.
- **Address mask request and reply** This type of messages is issued between hosts and routers to provide address mask to network hosts.
- **Router Solicitation and Advertisement** Router solicitation message is a broadcast or multicast message sent from hosts onto their networks to

discover which routers are alive and functioning. The routers respond with an advertisement broadcast or multicast message containing their information. Router advertisement messages can also be sent periodically by routers as some way of making network hosts aware of their presence.

It is clear from Table D.1 that the governing fields of ICMP packet types are; ICMP Type and ICMP Code, Figure D.5 illustrates ICMP Packet Format.

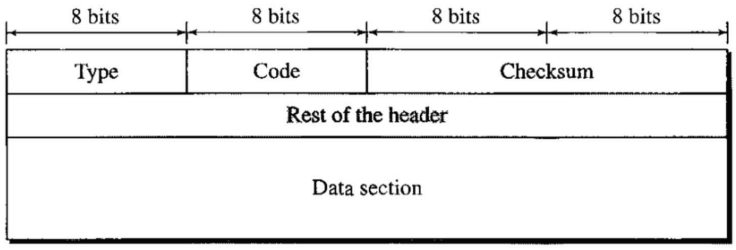


Figure D.5: ICMP Packet Format. [1]

From Table D.1 and Figure D.5, it is evidential that

Type ICMP Type is an 8 bit field that indicates the type of ICMP packet.

Code ICMP Code is an 8 bit field that expresses the reason for issuing that type of packet.

Checksum It is a 16 bit field that handles error detection of ICMP header.

Data In error packets, the data part carries the information of the original IPv4 packet within which the error occurs, in query packets it is empty.

Rest of the Header It contain fields that depend on the type of ICMP packet, in our case we focus only on Echo request and reply packets, in this case, the rest of the header consists of two 16 bit fields; Identifier and Sequence Number. They are responsible for matching echo requests to their respective echo replies.

D.5 Internet Group Management Protocol (IGMP)

IGMP helps multicast routers to create and keep track of updates for multicast groups which contain one or more members. Figure D.6 depicts the fields of IGMPv2 packet.

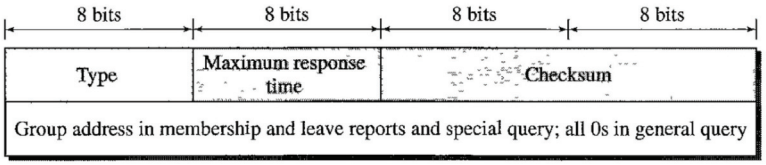


Figure D.6: IGMPv2 Packet Format. [1]

Type This is an 8 bit field that indicates the type of IGMP message; General or Special query (0x11 = 17), membership report (0x16 = 22), or leave report (0x17 = 23). Query messages can be either general or special query messages.

Maximum Response Time (MRT) It is an 8 bit field holding tenth of seconds values, it indicates the time within which a query should be answered.

Checksum It is a 16 bit that handles error detection of IGMP header.

Group Address A field that contains the multicast group ID (or multicast address) for special query, membership report, and leave report messages. A value zero indicates a general query message.

The names of IGMP messages indicate their respective function. A general query message is sent by multicast routers to know which groups have contributing members. A special query message works the same way but it asks about a specific group. Meanwhile, a group membership report message indicates that the sender wants to join the specified group, contrarily a leave report message

means that the sender wants to leave the specified group.

D.6 Transport Layer Ports

A specific process running on a host can use one or more service, Transport layer protocols handle packet delivery between multiple host processes. Since the port fields in both TCP and UDP are 16 bit long, they offer the capability to have any port number ranging from 0 to 65,535. The Internet Assigned Numbers Authority (IANA) [88] identify and reserve ranges of port numbers as;

- **System Ports Range (0-1023)**: Basic system applications
- **User Ports Range (1024-49151)**: Specific User-designed applications, including officially registered applications and unofficial applications
- **Dynamic Port Range (49152-65535)**: Transient port numbers, also known as Ephemeral ports, they are not assigned to any application and can be used freely by the host to initiate connections when needed.

D.7 Transmission Control Protocol (TCP)

Transmission Control Protocol (TCP) is a connection-based, reliable transport protocol [1][58][59], Figure D.7 demonstrates the fields of TCP packet.

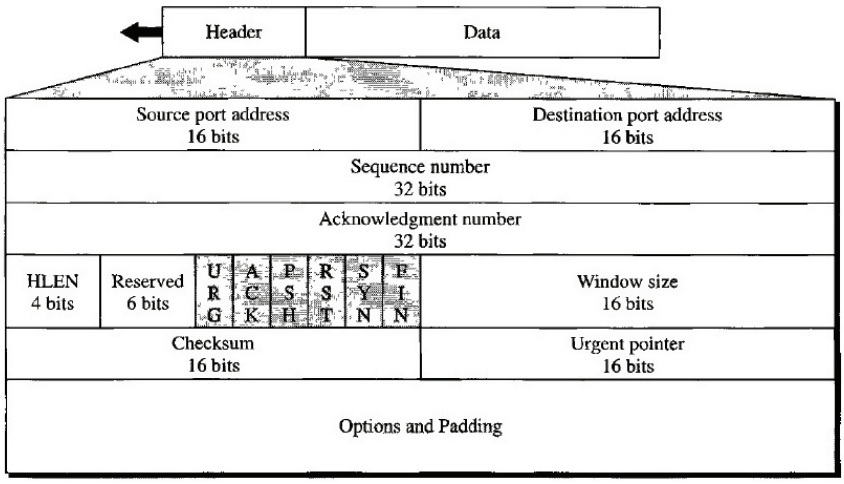


Figure D.7: TCP datagram Format. [1]

The description of TCP fields is:

Source Port It is a 16 bit field that indicates the port number of the service running on the host that is responsible for sending the packet.

Destination Port It is a 16 bit field that expresses the port number of the invoked service running on the destination host.

Sequence Number A 32 bit field that holds a number, assigned to the first byte within the packet.

Acknowledgment Number A 32 bit field that maintains a number indicating the byte number that source host expects to receive from the destination.

Header Length A 4 bit field that indicates the number of 4-byte (32 bit) words in the TCP header.

Reserved 6 bit field reserved for future use.

Control It is a field that includes 6 bit flags, one or more can be set for a given packet, that govern TCP connections between hosts.

Window Size A 16 bit field that designates the window size, number of bytes starting with the byte indicated by the *Acknowledgment Number*, that the source host can receive.

Checksum A 16 bit field used for error detection of the TCP packet.

Urgent Pointer A 16 bit offset from the sequence number that marks the existence of urgent data.

Options A variable length field that holds optional TCP header information.

D.8 User Datagram Protocol (UDP)

User Datagram Protocol (UDP) is a connectionless, unreliable protocol [1][60] that offers applications speed on the expense of reliability, Figure D.8 illustrates the fixed 8 byte header of UDP packets, also known as UDP datagrams.

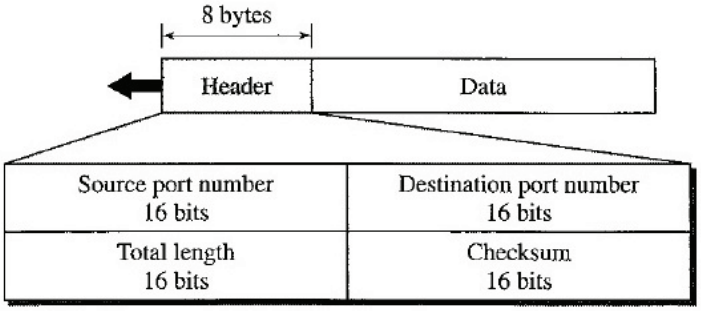


Figure D.8: UDP datagram Format. [1]

UDP packet fields are mainly:

Source Port A 16 bit field, like TCP, indicating the service running on the source host that is responsible for sending the packet.

Destination Port A 16 bit field, like TCP, that implies the intended service to be invoked on the destination host.

Length It is a 16 bit field that signifies the total length of the UDP packet (header + Data).

Checksum A 16 bit field used for error detection of the UDP packet.

D.9 Bootstrap Protocol (BOOTP)

BootP is used to supply a disk-less machine with the identity of the boot server and the boot file. Figure D.9 displays the fields of BOOTP header [1][61][62][63][64][65].

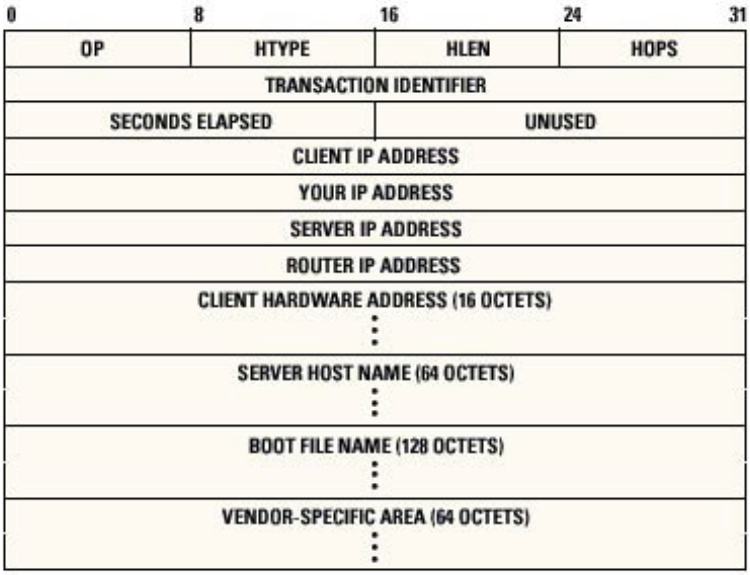


Figure D.9: BOOTP Packet Format. [4]

The interpretation of BOOTP fields is:

OP A 1 byte field that indicates the type of the packet; 1 is *bootrequest* and 2 is *bootreply*

HTYPE A is 1 byte field that expresses the hardware address type; *e.g.*, a value of 1 indicates 10mb Ethernet

HLEN A 1 byte field that demonstrates the length of the hardware address; *e.g.*, value 6 for 10mb Ethernet

HOPS A 1 byte field that indicates the number of packets hops. This field is not used by the client, but routers can optionally use it for cross-gateway routing.

Transaction Identifier (xid) A 4 byte random value used to match boot requests to their respective replies

Seconds Elapsed A 2 byte field that indicates the seconds that passed since the client started to request to boot.

Unused 2 byte Reserved and not used

Client IP Address A 4 byte field that indicates the client's address, this field is assigned by the client during a request if the client knows its IP address.

Your IP Address A 4 byte field that is assigned by the server to indicate the client's IP address, if the client's initial request did not contain its IP address.

Server IP Address A 4 byte field filled by the server during a reply which signifies the server's IP address.

Router IP Address A 4 byte field, also known as gateway IP Address, it is optionally used to indicate the IP address of the gateway router.

Client Hardware Address A 4 byte field assigned by the client to indicate its hardware address

Server Host Name A 64 byte field that can optionally be used to indicate the server name from which the client will select a boot file.

Boot File Name A 128 byte field that indicates the full path and name of the boot file on the server, that the client uses to boot.

Vendor-specific Area Optional 64 byte field that indicates information about vendor, it contains free format tagged sub-fields that provide guidance settings to the client.

D.10 Domain Name System (DNS)

DNS protocol provides name to address mappings. A client that needs to communicate with a host sends a DNS query to the DNS server in order to resolve the host name to its respective address. The DNS server replies with a DNS response that contains the IP Address of the queried host name. Figure D.10 illustrates the types of DNS packets.

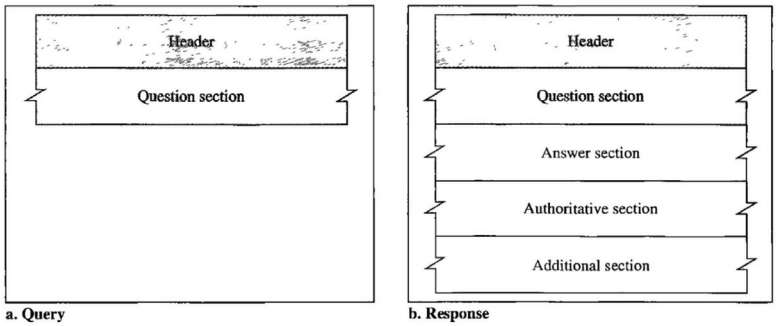


Figure D.10: DNS packet Types. [1]

As we can see in Figure D.10, DNS packets mainly contain a DNS header, DNS utilizes sections of information alongside the header which forms the structural body of DNS packets. Both packet types must contain DNS header and question section. DNS response packets contain 3 extra sections; Answer, Authority (or Authoritative), and/or Additional section. These sections are similar and can optionally be used according to the available information about the queried host. Each section consists of multiple information lists called Resource Records, *i.e.*, *RRs*, that hold information regarding the current query. Figure D.11 demonstrates the format of DNS Header.

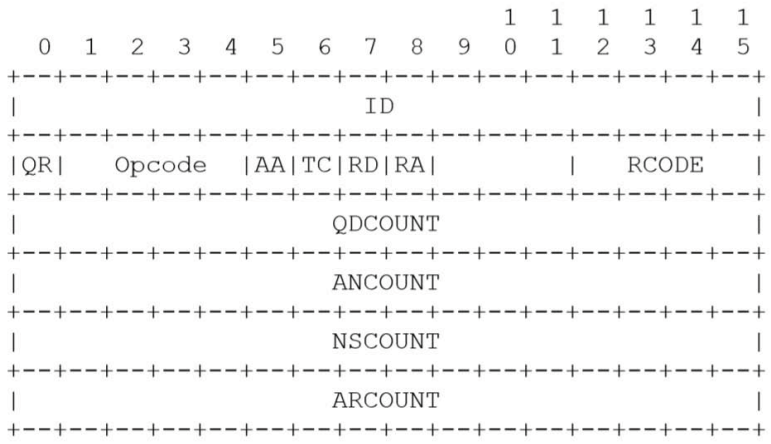


Figure D.11: DNS Header Format. [5]

The explanation of header fields indicated in the figure are:

ID A 16 bit identifier used to match DNS requests to their respective replies.

QR A 1 bit field that indicates the type of DNS packet; 0 query and 1 response.

OPCODE A 4 bit field whose value demonstrates the type of query according to the following values:

- Standard Query (QUERY): 0.

- Inverse Query (IQUERY): 1.
- Completion Query that include single (CQUERYU) or multiple (CQUERYM) answers: 2.
- Reserved: 4 – 15

AA 1 bit Authoritative Answer field, when it is set it implies that the responding DNS server is authoritative for the queried domain name.

TC 1 bit TrunCation field, when set it means that the packet is truncated because it contains more than 512 characters.

RD 1 bit Recursion Desired field, when set it instructs the DNS server to resolve the query in a recursive manner.

RA 1 bit Recursion Available field, it denotes whether recursive query support is available on the DNS server.

RCODE 4 bit Response Code field whose value indicates:

- 0: No Error Condition
- 1: Format Error
- 2: Server Failure
- 3: Name Error
- 4: Not Implemented
- 5: Refused
- 6 – 15: Reserved

QDCOUNT Unsigned 16 bit field that reveals the number of questions in the Question section.

ANCOUNT Unsigned 16 bit field that shows the number of RRs in Answer section.

NSCOUNT Unsigned 16 bit field that signals the number of Name Server RRs within the Authority section.

ARCOUNT Unsigned 16 bit field that specifies the number of RRs included in the Additional section.

Figure D.12 illustrates Question section format.

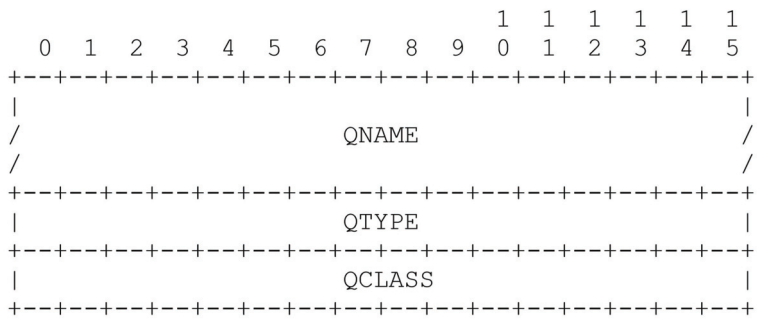


Figure D.12: DNS Question Section Format. [5]

The Question section mainly consists of *QNAME*, *QTYPE*, and *QCLASS*. *QNAME* is a variable length field which contains the name to be resolved. *QTYPE* is a 16 bit field that indicates the query type, *e.g.*, a decimal value of 1 means 'A' *i.e.*, host address. In other words, we want to know the IPv4 address of the queried name. *QCLASS* is a 16 bit field that designates the query class, *e.g.*, a decimal value of 1 implies 'IN' abbreviation for ARPA Internet. Table D.2 and Table D.3 illustrate QTYPE and QCLASS values respectively.

Type	Value	Meaning
1	A	A IPv4 Host Address
2	NS	An Authoritative Name Server
3	MD	A Mail Destination
4	MF	A Mail Forwarder
5	CNAME	The Canonical Name for an alias
6	SOA	The Start of a Zone of Authority
7	MB	A Mailbox domain name
8	MG	A Mail Group member
9	MR	A Mail Rename domain name
10	NULL	A null RR
11	WKS	A Well Known Service description
12	PTR	A Domain Name Pointer
13	HINFO	Host Information
14	MINFO	Mailbox or Mail List Information
15	MX	A Mail Exchanger record
16	TXT	Text record
28	AAAA	IPv6 Host Address
252	AXFR	Request for a Transfer of an Entire Zone of Authority
253	MAILB	Request for Mailbox-related records (MB, MG or MR)
254	MAILA	Request for Mail Agent RRs (MD and MF)
255	*	A request for all records

Table D.2: DNS Types

Class	Value	Meaning
1	IN	ARPA Internet
2	CS	Computer Science network (CSNET)

Table D.3: DNS Classes

Figure D.13 demonstrates the Resource Record (RR) format. A Resource Record consists of 6 fields: Name, Type, Class, TTL, RDLENGTH, and RDATA.

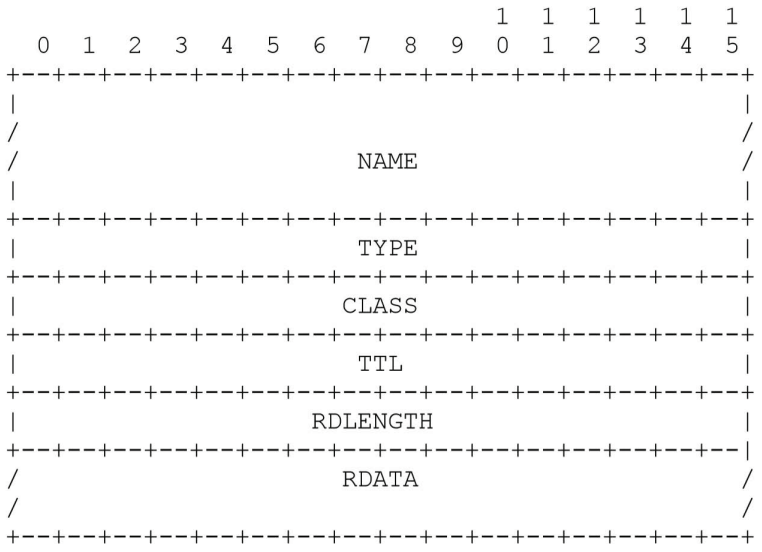


Figure D.13: DNS Resource Record Format. [5]

The *Name* field is a variable length compressed domain name associated with the current RR. *Type* and *Class* are 16 bit fields specifying the Type and Class of RDATA, their values are assigned according to tables D.2 and D.3. *TTL* is a 16 bit field that indicates the maximum valid time for RR. *RDLENGTH* is a 16 bit field that indicates the length in bytes of RDATA, while *RDATA* is a variable length field that holds the description of RR.

For example, if the Type is A (IPv4 host address) and the class is IN (ARPA Internet) then the RDATA will be an Internet address '*XXX.XXX.XXX.XXX*' of 4 bytes length. More details about Name Interpretation and RR types can be found in [5][66].

D.11 Link-Local Multicast Name Resolution (LLMNR)

LLMNR is considered a secondary name resolution protocol. The structure of LLMNR packets is the same as DNS packets regarding the sections and their related structures. Figure D.14 illustrates the format of LLMNR header.

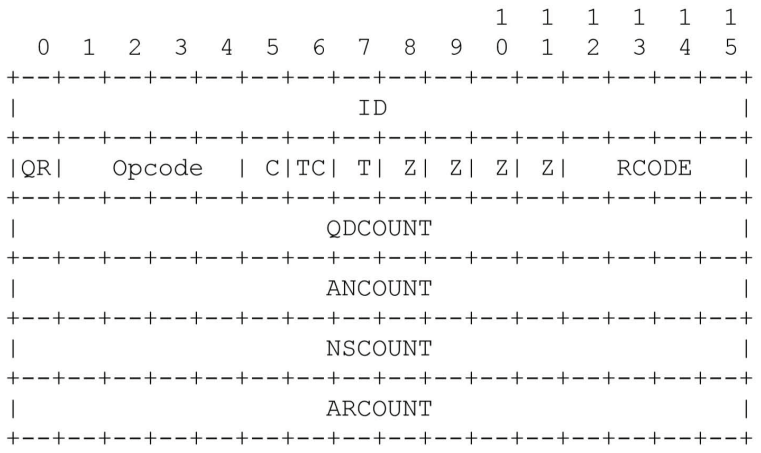


Figure D.14: LLMNR Header Format. [6]

The explanation of LLMNR header fields:

ID A 16 bit identifier that is used to match queries to their responses.

QR A 1 bit field that indicates the type of LLMNR packet; 0 LLMNR query, 1 LLMNR response

OPCode A 4 bit field that signifies the behaviour of queries and responses, further specifications may be defined for LLMNR OPCode.

C A 1 bit Conflict field, when set it implies that the sender of the query is confused about this specific query and that has not received a unique answer.

TC A 1 bit Truncation field indicating that the message is truncated due to its large size.

T A 1 bit Tentative field announcing that the server replying to the query is authoritative for the name but he has not assured its uniqueness yet.

Z 4 bits reserved for future use.

RCode 4 bit field Response Code

QDCOUNT unsigned 16 bit integer that signifies the number of entries in the Question section.

ANCOUNT unsigned 16 bit integer indicating the number of RRs in the Answer section.

NSCOUNT unsigned 16 bit integer implying the number of RRs in the Authority section.

ARCOUNT unsigned 16 bit integer showing the number of RRs in Additional section.

An LLMNR query can be multicast or unicast to resolve names. The main difference between DNS and LLMNR is the naming scheme. LLMNR resolves name to their respective addresses within the **IN-ADDR.ARPA** domain, *e.g., '52.0.2.10.IN-ADDR.ARPA'*.

D.12 NetBIOS Name Service (NBNS)

NBNS is another secondary name resolution protocol that can be used to handle multiple operations [70][7]. Figure D.15 illustrates the format of NBNS header.

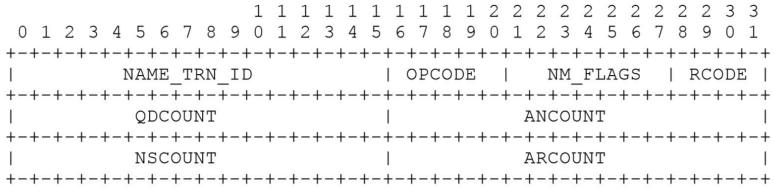


Figure D.15: NBNS header Format. [7]

NAME_TRN_ID 16 bit unique field used to match requests to their respective responses.

OPCODE A 5 bit field that indicates the type of packet, it consists of 2 fields: 1 bit RESPONSE Flag (R) field and 4 bits Opcode field. There values are interpreted according to:

- **R:** when set means that the packet is a response packet, else it is considered as query packet.
- **Opcode:** Operation Specifier: 0 = query, 5 = registration, 6 = release, 7 = WACK, and 8 = refresh.

NM_FLAGS A 7 bit field that consists of 5 flags and 2 reserved bits as followed (mentioned in order):

- **AA:** Authoritative Answer flag, it must be zero in queries. It means that the responding host is authoritative of the queried domain name.
- **TC:** Truncation flag, set to indicate that the size of the packet is large (greater than 576 bytes)
- **RD:** Recursion Desired flag, set only in queries which illustrates that the client requires a recursive name resolution within the domain.
- **RA:** Recursion Available flag, set in responses sent from NBNS servers only, which indicates that the server supports recursive resolution.

- **Reserved:** 2 bits reserved for future use
- **B:** Broadcast flag, 0 = unicast and 1 = multicast.

RCODE Result codes of request.

- **FMT_ERR:** 0x1 - Format Error
- **SRV_ERR:** 0x2 - Server failure
- **IMP_ERR:** 0x4 - Unsupported request error
- **RFS_ERR:** 0x5 - Refused error
- **ACT_ERR:** 0x6 - Active error
- **CFT_ERR:** 0x7 - Name in conflict error

QDCOUNT unsigned 16 bit integer that designates the number of entries in the Question section

ANCOUNT unsigned 16 bit integer that signifies the number of RRs in the Answer section

NSCOUNT unsigned 16 bit integer that implies the number of RRs in the Authority section

ARCOUNT unsigned 16 bit integer that specifies the number of RRs in the Additional section

NBNS does not use the same TYPE and CLASS values in Question section as DNS and LLMNR it uses the values in Table D.4

	Symbol	Value	Description
QTYPE	NB	0x0020	NetBIOS general Name Service Resource Record
	NBSTAT	0x0021	NetBIOS NODE STATUS Resource Record
QCLASS	IN	0x0001	Internet class

Table D.4: NBNS QTYPE and QCLASS values

Also, NBNS uses TYPE and CLASS values for RRs listed in Table D.5

	Symbol	Value	Description
RR_TYPE	A	0x0001	IP address Resource Record
	NS	0x0002	Name Server Resource Record
	NULL	0x000A	NULL Resource Record
	NB	0x0020	NetBIOS general Name Service Resource Record
	NBSTAT	0x0021	NetBIOS NODE STATUS Resource Record
RR_CLASS	IN	0x0001	Internet class

Table D.5: NBNS RRs TYPE and CLASS values

NBNS packets have multiple types according to the values of their header fields, which enables them to fulfill multiple functionalities:

1. Name Registration Request
2. Name Overwrite Request And Demand
3. Name Refresh Request
4. Positive Name Registration Response
5. Negative Name Registration Response
6. End-Node Challenge Registration Response
7. Name Conflict Demand
8. Name Release Request And Demand
9. Positive Name Release Response
10. Negative Name Release Response
11. Name Query Request
12. Positive Name Query Response
13. Negative Name Query Response

14. Redirect Name Query Response
15. Wait For Acknowledgement (Wack) Response
16. Node Status Request
17. Node Status Response

D.13 Simple Network Management Protocol (SNMP)

SNMP utilizes the concept of manager and agents, in which manager (or management station) communicates with agents. It manages network elements by; minimizing the complexity of management functions, upgrading flexibility of monitor and control paradigm, and independency of management architecture [1][8][71][72].

SNMP uses two other implicit protocols to achieve management tasks; Structure of Management Information (SMI) and Management Information Base (MIB). Since the management body is independent of the network and host structure, a unified set of rules has to be identified to facilitate the management process, SMI protocol defines these rules. SMI defines rules for naming objects, object types, and encoding. MIB, on the other hand, defines management entities of hosts regarding their objects, object types, and object relationships. On top of these protocols, according to SMI rules and MIB definitions, SNMP defines the appropriate packet formats and structures. SNMP packets allow management information to be exchanged by reading objects statuses and modifying their values .

Abstract Syntax Notation One (ASN.1) defines a unique set of data types that is independent, some of them are simple data types and others are complex, Table D.6 illustrates a sample of those data types and their respective values.

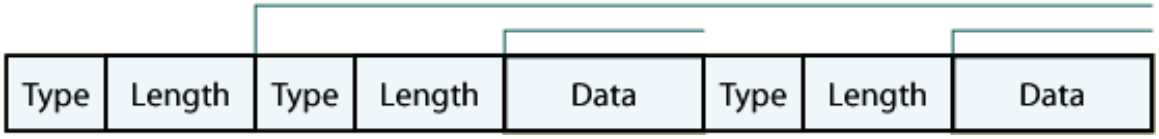
Primitive Data Types	Value	Complex Data Types	Value
Integer	0x02	Sequence	0x30
Octet String	0x04	GetRequest PDU	0xA0
Null	0x05	GetResponse PDU	0xA2
Object Identifier	0x06	SetRequest PDU	0xA3

Table D.6: ASN.1 sample Data Types [8].

Where PDU is an abbreviation for Protocol Data Unit, units of data that SNMP uses for communication between managers and agents.

Basic Encoding Rules (BER) states that information has to be structured within packets in the form of triplets containing: Name, Length, and Value, Figure D.16.

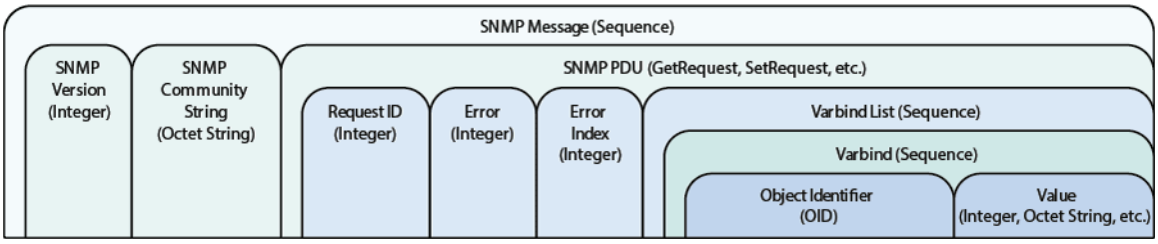
(a) BER Encoded Primitive Data Type.



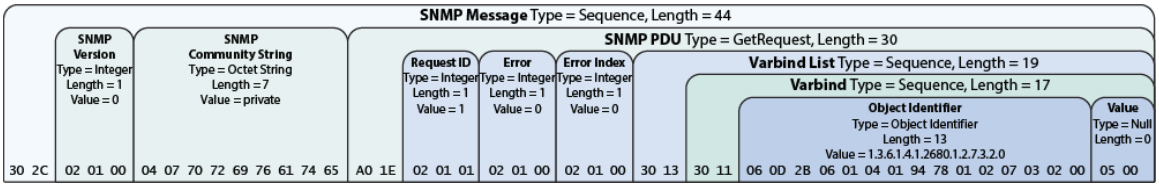
(b) BER Encoded Complex Data Type.

Figure D.16: BER Encoded Fields. [8]

Figure D.16 illustrates the BER encoding of both primitive (Figure D.16a) and complex (Figure D.16b) SNMP data types. Now that we have explained all this about SNMP let's take a look at SNMP packet format, Figure D.17. Figure D.17a illustrates SNMP packet format and Figure D.17b shows an example with values.



(a) SNMP Packet.



(b) SNMP Example.

Figure D.17: SNMP Packet Format. [8]

SNMP Packet fields are mainly:

SNMP message: A sequence that represents the whole SNMP packet.

SNMP Version: An Integer that identifies the version of SNMP

SNMP Community String: A variable length byte String that adds security to SNMP devices, '*public*' community string means read-only community, while '*private*' means read-write community.

SNMP PDU: An SNMP PDU that represent the body of SNMP packet

Request ID: An Integer that matches SNMP requests to their responses

Error: An integer that represents the type of error. Its value includes:

- **0x00** – No error occurred
- **0x01** – Response message too large
- **0x02** – Name of the requested object was not found

- **0x03** – Data type in the request did not match the data type in the SNMP agent
- **0x04** – SNMP manager attempted to set a read-only parameter
- **0x05** – General Error, some error that is not listed in the above values.

Error Index: An Index that points to the Object that caused the error.

Varbind List: A Sequence of Varbinds.

Varbind: A Sequence of two fields, an Object ID and the value for/from that Object ID.

Object Identifier: An Object Identifier that points to a specific parameter in the SNMP agent, for more details about how to understand this field, please refer to [89].

Value
- SetRequest PDU: Value is assigned to the specified OID of the SNMP agent.
- GetRequest PDU: Value is a Null that acts as a placeholder for the return data.
- GetResponse PDU: The returned Value from the specified OID of the SNMP agent.

D.14 Simple Service Discovery Protocol (SSDP)

SSDP header format include 3 main containers of information:

1. SSDP Start-line
2. SSDP Message Header fields

3. SSDP Header field extensions

According to the type of SSDP packet the format will remain the same but with different fields in each container. SSDP packet types include:

1. Advertisement
    a. Device available – NOTIFY with ssdp:alive
    b. Device unavailable – NOTIFY with ssdp:byebye
    c. Device Update - NOTIFY with ssdp:update
2. Search
    a. Search request with M-SEARCH
    b. Search response

D.14.1 SSDP Advertisement: Device Available

SSDP *Advertisement* packets are used to advertise the state of a device concerning its services and root devices. In such case, SSDP packets must be multicast using standard address and port combination '*239.255.255.250:1900*', the source and destination ports of such packets will be equal to *1900*.
Figure D.18 illustrates the *SSDP Device Available* packet format.

```
NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
CACHE-CONTROL: max-age = seconds until advertisement expires
LOCATION: URL for UPnP description for root device
NT: notification type
NTS: ssdp:alive
SERVER: OS/version UPnP/1.1 product/version
USN: composite identifier for the advertisement
BOOTID.UPNP.ORG: number increased each time device sends an initial announce or an update
message
CONFIGID.UPNP.ORG: number used for caching description information
SEARCHPORT.UPNP.ORG: number identifies port on which device responds to unicast M-SEARCH
```

Figure D.18: SSDP Device Available Packet Format. [9]

We explain the meaning of packet fields and use example values from our captured traffic (if exists):

Request Line (Must): It must be "$NOTIFY * HTTP/1.1$". '*NOTIFY*' is a general method for sending notifications and events, and '$*$' generally emphasize any resource upon the device. '*HTTP/1.1*' indicates the version of HTTP used within the packet.

HOST (Required): It is the standard multicast address and port number '*239.255.255.250:1900*'.

CACHE-CONTROL (Required): This is an integer field that designates the duration (in seconds) within which this availability packet is valid before it expires.

LOCATION (Required): It holds a URL for the description of the advertised resource on the device, *e.g., http://192.168.10.50:50002/*

NT (Required): Notification Type of the packet, *e.g., urn:schemas-upnp-org:service:RenderingControl:1*

NTS (Required): Notification Sub Type of the packet, *e.g., ssdp:alive*

SERVER (Required): Specifies the type of server that provides the resource, *e.g., WINDOWS, UPnP/1.0, Intel MicroStack/1.0.1497*

USN (Required): Unique Server Name,
e.g., uuid:3855B43B-A276-0C30-B22C-7FA461B2FC9D::urn:schemas-upnp-org:service:Re

BOOTID.UPNP.ORG (Required): This field represents the boot instance of the advertised resource

CONFIGID.UPNP.ORG (Required): This field indicates the configuration number of the advertised resource.

SEARCHPORT.UPNP.ORG (Optional): Mainly, this field is used to designate a different port number other than 1900, to help in unicast communication of Search packets.

When the duration of the availability packet expires the device has to renew availability of its resource with another *Device Available* packet.

D.14.2 SSDP Advertisement: Device Unavailable

If the status of the resource or the device changes and results in their removal from the network, the device has to send an unavailable notification packet that tells the network of such change. Figure D.19 demonstrates the format of such packet.

```
NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
NT: notification type
NTS: ssdp:byebye
USN: composite identifier for the advertisement
BOOTID.UPNP.ORG: number increased each time device sends an initial announce or an update
message
CONFIGID.UPNP.ORG: number used for caching description information
```

Figure D.19: SSDP Device Unavailable Packet Format. [9]

The fields of *SSDP Device Unavailable* packet are the same as those of *Device Available* packet. The only difference is that *NTS* field contains *ssdp:byebye*.

D.14.3 SSDP Advertisement: Device Update

When the change in resource does not result in removal from the network, *i.e.*, the resource is still available but in this case it takes a new boot instance, therefore,

its new boot instance number has to be announced. This announcement is made by sending a Device Update Packet, Figure D.20.

```
NOTIFY * HTTP/1.1
HOST: 239.255.255.250:1900
LOCATION: URL for UPnP description for root device
NT: notification type
NTS: ssdp:update
USN: composite identifier for the advertisement
BOOTID.UPNP.ORG: BOOTID value that the device has used in its previous announcements
CONFIGID.UPNP.ORG: number used for caching description information
NEXTBOOTID.UPNP.ORG: new BOOTID value that the device will use in subsequent announcements
SEARCHPORT.UPNP.ORG: number identifies port on which device responds to unicast M-SEARCH
```

Figure D.20: SSDP Device Update Packet Format. [9]

Once more the meaning of these fields are the same as *Device Available SSDP* packet, with the difference that the *NTS* field contains *ssdp:update*.

D.14.4 SSDP Search request with M-SEARCH and Search Response

SSDP search messages include requests and responses. SSDP allows devices to search for specific resource on the network. The search procedure is done by sending a request packet about the required resource, then the device that owns this resource and it is available, responds in a unicast manner. The source port of the requestor is a dynamically assigned port not 1900 like in advertisement packets, but the destination port is 1900. Figure D.21 represents the format of SSDP search request.

```
M-SEARCH * HTTP/1.1
HOST: 239.255.255.250:1900
MAN: "ssdp:discover"
MX: seconds to delay response
ST: search target
USER-AGENT: OS/version UPnP/1.1 product/version
```

Figure D.21: SSDP M-Search Request Packet Format. [9]

The meaning of these fields is:

Request Line (Must): It must be "$M-SEARCH*HTTP/1.1$". '*M-SEARCH*' is a method for search requests, and '$*$' generally emphasize any resource upon the device. '*HTTP/1.1*' indicates the version of HTTP used within the packet.

HOST (Required): Its meaning is the same as before. The difference is, with multicast requests the standard address and port is used '*239.255.255.250:1900*'. On the other hand, when the request is unicast the used address format is *hostname:portNumber*.

MAN (Required): It defines the scope of HTTP extension framework, it must be set to *"ssdp:discover"*

MX (Required): It specifies a duration in seconds in a range from 1 to 5 that the responder has to wait before processing the request.

ST (Required): Search Target URI, *e.g., upnp:rootdevice*

USER-AGENT (Optional): string field that is specified by the server in the format *"OS name/OS version/product name/product version"*. *e.g., "USER-AGENT: unix/5.1 UPnP/1.1 MyProduct/1.0"*

SSDP response packets incorporates fields from both; SSDP requests and SSDP Device Available packets, Figure D.22.

```
HTTP/1.1 200 OK
CACHE-CONTROL: max-age = seconds until advertisement expires
DATE: when response was generated
EXT:
LOCATION: URL for UPnP description for root device
SERVER: OS/version UPnP/1.1 product/version
ST: search target
USN: composite identifier for the advertisement
BOOTID.UPNP.ORG: number increased each time device sends an initial announce or an update
message
CONFIGID.UPNP.ORG: number used for caching description information
SEARCHPORT.UPNP.ORG: number identifies port on which device responds to unicast M-SEARCH
```

Figure D.22: SSDP M-Search Response Packet Format. [9]

The additional fields that were not explained before are:

DATE (Recommended): A field that specifies the date of generating the response.

EXT (Required): A field used for backward compatibility with UPnP 1.0.

All the remaining fields are the same as explained before.