Towards Graph-Based Intrusion Detection in Cybersecurity

by

Etienne Pieter van de Bijl

A thesis submitted in partial fulfillment of the

requirements for the degree

Master of Science

Supervisors:	*Graduation supervisor:*	*Second reader:*
Jan Klein MSc	Prof. dr. Rob van der Mei	dr. Mark Hoogendoorn
Joris Pries MSc		

Ministerie van Binnenlandse
Zaken en Koninkrijksrelaties
Turfmarkt 147
2511 DP Den Haag

Centrum Wiskunde &
Informatica
Science Park 123
1098 XG Amsterdam

Vrije Universiteit Amsterdam
Faculty of Science
De Boelelaan 1085
1081 HV Amsterdam

March 2020

Abstract

Intrusion prevention systems are the second line of defence in computer security. These systems aim at detecting malicious activities by monitoring computer and/or network traffic. At the heart of these systems are the intrusion detection systems, which aim at detecting malicious activities. Nowadays, there is a preference for detection systems aiming at known intrusions. This is a result of a study gap that exists on finding previously unknown intrusions.

Anomaly detection is the task of finding observations that does not confirm to a certain normal or expected behaviour. One advantage of this technique is that it is able to find new intrusions. In the field of anomaly detection, there is a growing interest in anomaly detection techniques using the underlying graph-structure in the dataset. Only considering the communication behaviour between two devices might bypass correlations between other hosts. Finding deviating network changes could indicate that the network is under attack.

This research focusses on developing an intrusion detection system that utilizes a graph-based approach in combination with an anomaly detection algorithm from the Machine Learning field. Connection data are transformed to anomaly scores and these scores are used as edge weights between hosts. The edges indicate the likeliness of the behaviour seen over the network. By tracking the changing dynamic of the network with these edge weights, deviating network changes are flagged.

Several contributions to the cybersecurity field are presented in this thesis. First of all, new and improved labelling schemes are constructed for two of the three selected datasets. The datasets selected in this research are the CIC-IDS-2017, the ISCX-IDS-2012 and the UNSW-NB15. Secondly, in these datasets it is tested if indeed anomalous traffic is different from normal traffic by applying multiple supervised learning techniques on the dataset. Thirdly, anomaly detection techniques, such as principle component analysis and isolation forest, are applied on connection level to study the performance of these methods on connection level. Lastly, four unweighted and four weighted distance metrics between graphs are compared on whether or not they capture malicious network change. In the weighted metrics, anomaly scores of the anomaly detectors are used as edge weights.

The results of this study show that on connection level the supervised learning techniques are able to distinguish normal behaviour from malicious behaviour when more protocols are used to detect intrusions. Furthermore, detecting malicious network change in the UNSW-NB15 dataset by using the weighted Umeyama distance or the unweighted MCSVD had a F_1 score of at least 0.95 for the HTTP and UDP datasets.

Preface

This graduation thesis is the result of my internship at the Ministry of the Interior and Kingdom Relations and the Centre for Mathematics and Computer Science (CWI). The internship is part of the Master Project Business Analytics, which is the final course of the master Business Analytics at the Vrije Universiteit Amsterdam. The course requires students to do a six month internship at an external organization. The aim of the internship is to conduct scientific research and to establish collaborations between science and businesses. This project should have practical and relevant value for the external organization, as well as scientific value for the university. The goal for the student is to develop deeper knowledge, understanding and capabilities in the field of Business Analytics in practice, research and science.

The hosting organizations of this internship have specified their interest in conducting research in the field of cybersecurity. Research developments in this field are crucial for keeping computer security up-to-date. This can be accomplished by developing and producing new methods, which could also be used in different domains.

Conducting a research project successfully can not be possible without the support of other people. First, I would like to thank my direct supervisors Jan Klein MSc and Joris Pries MSc, who guided me through the internship. Their guidance, expertise and daily supervision had a positive influence on this thesis. They gave me the right insights on how to conduct scientific research. I also would like to thank prof. dr. Rob van der Mei who supported me and motivated me to leave no stone unturned. Furthermore, I am grateful to the involved members at the Ministry and the CWI who gave me the opportunity to conduct this research. Another involved party in the research was the Ministry of Defense. Raymond Hinfelaar give me useful insights in the cybersecurity domain. At last, my family and friends have always supported me during this period. I truly appreciate their advises and counseling.

Etienne Pieter van de Bijl
March 2020

Contents

Chapter 1

Introduction

On 27 June 2017, one of the most devastating cyberattacks, called NotPetya, infected thousands of PCs around the world. This malware primarily targeted digital systems in Ukraine, but the cyberattack escalated beyond this target and infected countless machines all around the world. The name of this cyberattack originates from the resemblance to the ransomware Petya (2016), which is the Russian translation of malware. Petya was a piece of code which denied access to a computer by encrypting the system until a ransom was paid. Unlike Petya, any ransom payment was pointless in the NotPetya malware, as the computer's contents were still shattered beyond repair. The cyberattack was able to spread rapidly over the Internet due to a vulnerability in a widely used accounting software. In Ukraine, the malware infected at least four hospitals, six power companies, two airports, more than 22 Ukrainian banks, ATMs, card payment systems and almost every federal agency. The global damage by NotPetya is estimated to be more than 10 billion US dollars in total (Greenberg, 2018).

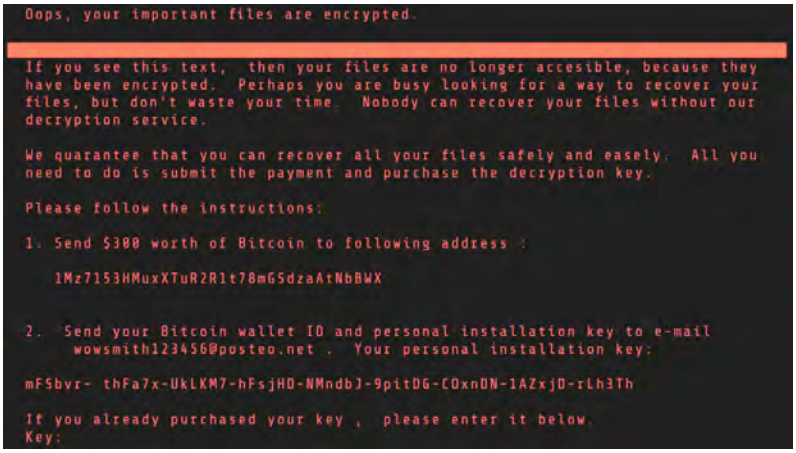


Figure 1.1: Ransom note displayed by a computer infected with NotPetya (cbronline, n.d.)

The NotPetya cyberattack shows the potential disastrous consequences when software contains vulnerabilities which can be misabused by attackers. When security measures fail to detect the exploitation of these vulnerabilities, attackers are clear to execute harmful activities without detection. Unfortunately, defenders have a disadvantage over attackers as a defender must defend all parts of a computer system, while an attacker only has to find one exploitable weak part to exploit (Howard & Leblanc, 2002). Another disadvantage for the defender is that cyberattacks have become profitable, easily accessible and that there is a low risk in getting prosecuted (NCTV, 2018). In summary, cybersecurity and cybercrime are dynamical fields where attacker and defender try to outsmart each other and security measures should be up-to-date. Therefore, it is necessary to

further develop security systems to detect and prevent these attacks or intrusions from happening.

To prevent attacks from happening, researchers in computersecurity have developed multiple lines of defence. *Firewalls* are often referred to as the *first line of defence* (David, 2017; Sisco, 2018). A firewall determines with a set of rules which traffic is rejected and which is not. These rules can, however, be circumvented by the attacker, as was seen in the NotPetya attack where trusted software was misused. For that reason, computer security researchers have proposed another line of defence: an Intrusion Prevention System (IPS) (Xinyou Zhang, Chengzhong Li, & Wenbin Zheng, 2004). This system aims at detecting malicious activities by monitoring computer and/or network traffic. When such activities are detected, the IPS attempts to block or stop these intrusions. Within IPS, an Intrusion Detection System (IDS) is performing the task of detecting cyberattacks (Axelsson, 2000; Denning, 1987).[1] While intrusion prevention is the end goal, detecting intrusions is the heart of preventing them.

The two main intrusion detection methods in IDS are *signature detection* and *anomaly detection* (Axelsson, 2000). In signature detection, intrusions are detected by using precise descriptions of known malicious activities, while anomaly detection systems utilize a notion of normal activities and flag behaviour which deviates from this notion. Anomaly detectors are more useful for finding previously unknown attacks. In contrast, signature detectors are better for detecting known intrusions. These methods have been extensively studied over the last years, but there is a striking imbalance in their deployments. There is a general preference for using signature detectors over anomaly detectors. One major challenge which limited anomaly detectors to be used in practice is a lack of up-to-date evaluation datasets (Sommer & Paxson, 2010). Often used intrusion detection datasets, such as the KDD99 dataset and its successor NSL-KDD, are outdated and heavily criticized to be used for accurate evaluation (Thomas, Sharma, & Balakrishnan, 2008). Over the years, the research community responded to this lack by proposing new datasets for intrusion detection.

Another development in the field of anomaly detection is growing interest in creating new anomaly detection methods for datasets with an underlying network structure (Akoglu, Tong, & Koutra, 2015). Data objects might have interdependencies which can be taken into account by using a graph approach. A graph approach is a powerful means to show relational dependencies between objects, which is not taken into account while assuming that all data points are independent (Akoglu et al., 2015). In the intrusion detection setting, monitoring hosts individually might for example bypass communication behaviour correlations between the hosts. Therefore, tracking the changing nature of the network traffic could show malicious activities.

Problem Statement

While anomaly detection methods using a graph-based approach have great potential, they have not been applied on the latest developed intrusion detection datasets. These datasets are primarily connection-oriented and the anomaly detection techniques which have been applied on them are mainly machine learning algorithms. Machine learning algorithms are very popular nowadays and have proven to be very accurate in many practical applications. These two approaches are studied separately, but applying a combination of these methods on the latest intrusion detection datasets has not been done before. Therefore, it could be of scientific and practical interest to study the performance of machine learning anomaly detectors in combination with a graph-based detector.

[1] In contrast to other literature, we see the task of detecting intrusions as a part of the task of preventing an intrusion. Therefore, we see IDS as part of IPS and not as two non-overlapping systems.

Scope of this Research

The *aim of this research* is to develop an intrusion detection algorithm which utilizes a graph-based approach in combination with a machine learning algorithm. This implies that we are working with network intrusion detection systems. The goal is to translate connection-level behaviour to anomaly scores using machine learning. These anomaly scores are used as edge weights for the graph-based anomaly detector. These scores indicate the likeliness of traffic seen between two hosts. The connection-level behaviour is transformed into anomaly scores by using two algorithms: Principal component analysis and Isolation Forest. One assumption which is tested before applying these models is: *"Malicious behaviour differ from normal behaviour"*. Checking this assumption is essential when applying anomaly detection techniques as intrusion detection system. To test this assumption, several supervised learning techniques are used to observe whether these models are able to distinguish malicious from normal traffic. After translating to anomaly scores, the graph-based anomaly detector logs the change of the network over time and tries to detect abnormal changes in the network which could indicate malicious behaviour seen on the network. Several distance metrics are studied to determine which metric is most suitable to detect abnormal change on the network.

Where most researches only apply developed detector(s) to one dataset, we are interested in applying the detector to several datasets to study the performance of our detector on different networks. The datasets of interest are the CIC-IDS-2017, the ISCX-IDS-2012 and the UNSW-NB15. To make the evaluations of the method between the datasets comparable, a tool named Zeek is used to extract information from the provided raw Internet traffic to create protocol specific datasets. Zeek logs events observed on the network, the transport and the application layer and stores all gathered data in predefined protocol specific log files. Instead of limiting to the events observed on the network layer, information on specific protocol usages are also used for the task of intrusion detection in this research.

Structure of the Thesis

The thesis is structured as follows. Chapter 2 provides some of the relevant basics of computer networking which is useful to help understanding the information sent between devices. Chapter 3 discusses characteristics of intrusion detection systems and provides advantages and disadvantages of different intrusion detection methods and technologies. It helps the reader to put the proposed method in the framework of all available intrusion detection techniques. Chapter 4 states relevant studies that show the state of the art on anomaly detection in cybersecurity to make comparisons with this research. Chapter 5 states the experimental setups of the three selected datasets. Several inherent problems with the provided datasets are discussed and some solutions are given to overcome these problems. The raw Internet traffic are translated with Zeek into protocol specific log files. As the features are not directly applicable for machine learning algorithms, Chapter 6 shows how the generated log files are prepared for machine learning purposes. Chapter 7 shows an analysis of the datasets and describes important aspects of the data. In Chapter 8, the used methods to test the assumption that malicious traffic is indeed different than normal traffic, the algorithms to transform connection-level behaviour in anomaly scores and the graph distances metrics that show the change of the network are described. Chapter 9 shows the results the applied methods on the discussed datasets. Finally, Chapter 10 gives the discussion and conclusion of this research.

Chapter 2

Computer Networking

The Internet is one of the largest and most complex systems ever created by mankind. It connects approximately 48% of the world's population (ITU, 2017) with more than 17 billion devices (Leuth, 2018). These statistics show the enormous scale of this system, but what are the underlying principles that make this system work? Before diving in the intrusion detection methods, it is useful to know some fundamental principles in computer networking as they can help understanding the information sent between devices. This chapter is used as a comprehensive overview on the principles and structures of computer networking and states relevant terminology which is used throughout this thesis.

2.1 Preliminaries

The internet is a system of interconnected *computer networks* which connects devices all over the world. The term computer network seems outdated as not only desktop PCs are connected to this system, but also other devices, such as smartphones, tablets and televisions. These devices are called *hosts* or *end systems*. There are two paradigms in which hosts can communicate: the *client-server* architecture and the *peer-to-peer* (P2P) architecture. These paradigms are important to distinguish as they have different characteristics in the flow of information between hosts. In the client-server architecture, a *server* provides services requested by one or more *clients*, such as sending web pages. In the P2P architecture, information between hosts is transferred without the use of a dedicated server. Hosts are both client and server in this framework. The blockchain technology is one example of P2P communication (Kurose & Ross, 2012).

Hosts are connected by a network of *communication links* and *packet switches*. Information is sent from one host to another via these links, which are either wireless or wired. Computer networks can be seen as a network of highways and roads over which information is sent from one place to another. This network is used by many independent trucks which carry packets containing parts of the sent data. Packet switches, often routers and link-layer switches, are devices which forward packets to the ultimate destination. According to Tanenbaum and Wetherall (2011), there are two network characteristics which can be used to distinguish networks: transmission technology and scale. In transmission technology, packets are sent either using *unicasting*, in which there is exactly one sender and exactly one receiver, and *broadcasting* (or *multicasting*), where packets are sent to all receivers in the network but only the intended receiver will respond. As the distance between hosts differ, different transmission technologies (e.g. physical media) are required to send packets. Figure 2.1 shows the scale sizes of different networks. The scale of the network determines the range of hosts which are directly connected to the network. Local Area Networks (LAN) is the network scale of interest in this thesis as this is the cover range of most intrusion detection systems.

Interprocessor distance	Processors located in same	Example
1 m	Square meter	Personal area network
10 m	Room	Local area network
100 m	Building	
1 km	Campus	
10 km	City	Metropolitan area network
100 km	Country	Wide area network
1000 km	Continent	
10,000 km	Planet	The Internet

Figure 2.1: Network ranges (Tanenbaum & Wetherall, 2011)

2.2 Network Architecture

Hosts are able to communicate with each other by sending packets travelling through one or multiple networks. Communication between end systems is governed by *protocols*. To understand the notion of a protocol, consider the following human analogy. Two humans are introduced to each other. Do they kiss each other on the cheek? Do they give each other a hand? When both parties are not willing to do the same, communication will be difficult, if not stopped. Only when both parties agree on the communication conditions they can start sharing information. In a more formal way, a protocol is defined as: "*... the format and order of messages exchanged between two or more communicating entities, as well as the actions taken on the transmission and/or receipt of a message or other event*" (Kurose & Ross, 2012).

As there are many protocols defined for internet communication, network designers have structured these protocols in stacked layers. The idea behind the stacked system is that each of these layers provide a service to the layer on top of it. Figure 2.2 illustrates the communication of two hosts using five layers of protocols. At the bottom, the physical medium is the layer in which the actual communication occurs. Each layer on top is used by a protocol which provides services to protocols in the layer above it. In the interface level, the operations and services which are available to the upper layer can be observed.

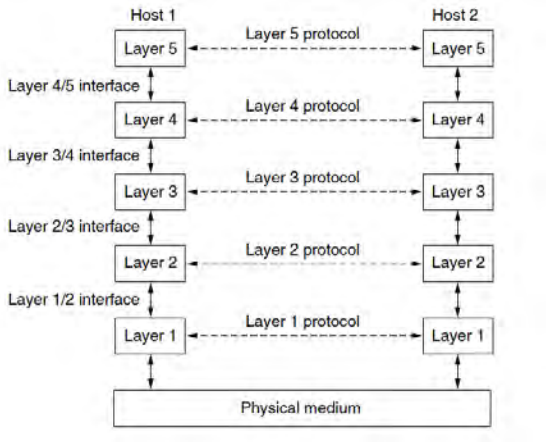


Figure 2.2: Layers, protocols and interfaces (Tanenbaum & Wetherall, 2011)

Layers can offer two types of services to the layers on top of them: *connection-oriented* and *connectionless*. The connection-oriented service has similarities with the telephone system. Connections

need to be established before they can be used and when the service is no longer required, the connections are terminated. In contrast, a connectionless service is modelled as the postal system. Each message has a destination address and will be routed through the system to this destination without the requirement of establishing a connection.

Services can be characterized by their reliability. They are either *reliable* or *unreliable*. Using a reliable service implies that data will arrive in the intended format at its destination. Furthermore, in a reliable service, the receiver will inform the sender that the messages are properly received. In contrast, in an unreliable service there is no guarantee that data will actually arrive. A reliable connection-oriented service seems optimal, but there are situations in which this is not desirable. Table 2.1 gives several examples in which other service characteristics are more suitable. While downloading a movie, it is important that each part of the movie is correctly sent to the receiver. In contrast, when making a video call over the internet, such as a Skype video conversation, getting the latest data is more relevant. With real-time applications real-time information is required. In these situations, one does not want to get data that was sent some time ago. Some applications do not require connections to be established as the establishment of a connection takes time and getting the latest information is more important than receiving it in a reliable manner.

Table 2.1: Types of services requirements

Service	**Data Delivery Guarantee**	**Example**
Connection-oriented	Reliable	Movie downloading
Connection-oriented	Unreliable	Voice over IP
Connectionless	Reliable	Text messaging
Connectionless	Unreliable	Junk mails

2.3 Reference Models

As discussed in Section 2.2, the network architecture can be structured in layers. Researchers have proposed several models to structure layers and assign protocols to these layers. These models are the so-called reference models of the internet. Table 2.2 illustrates some of the proposed models and their proposed layer scheme. The Open Systems Interconnection (OSI) model (Day & Zimmermann, 1983) is one of the first steps in the standardization of protocols (Tanenbaum & Wetherall, 2011). This model distinguishes seven layers in which the protocols can be assigned. The TCP/IP stack has only four layers as it combines the session, presentation and application layer in one layer and it does not include the physical layer. Two other reference models have been proposed by Tanenbaum and Wetherall (2011) and Kurose and Ross (2012). While these two reference models look similar, the creators define different purposes to the layers.

Table 2.2: Reference models and their differences

Layer	**OSI Model**	**TCP/IP model**	**Tanenbaum**	**Kurose**
7	Application			
6	Presentation			
5	Session		Application	Application
4	Transport	Application	Transport	Transport
3	Network	Transport	internet	Network
2	Data link	internet	Link	Link
1	Physical	Link	Physical	Physical

In this research, the network layer, transport layer and application layer are of interest for the task of intrusion detection. Therefore, we will only discuss these layers in the following sections. For

the physical layer and data link layer it is relevant to know that these layers provide the service of sending bits/frames from one host to the other. Finding intrusions on these layers is beyond the scope of this thesis.

2.3.1 Network Layer

The network layer is in between the data link layer and the transport layer. This layer provides services to the transport layer. The main function in this layer is routing *packets* from source to destination. This layer must know the topology of the network (links and routers) in order to send packets through the network in such a way that packets reach their final destination. In the network layer, each internet host or entity is identified with a network layer address.

internet Protocol

The main protocol in the network layer is the *internet Protocol* (IP). This protocol allows applications to carry out point-to-point communications. Nowadays, there are two versions available and in use: IP version 4 (IPv4) (Postel, 1981a) and IP version 6 (IPv6) (Deering & Hinden, 1998). The difference between these two versions is that the IPv4 address has a fixed size of 32 bits, while IPv6 addresses are 128 bits long. The IPv4 32-bit addresses are often represented as four digits separated by dots. So for example 1.2.3.1 corresponds to 00000001 00000010 00000011 00000001 in binary representation. With this representation, there are in total 2^{32} different unique addresses possible, which is around 4.3 billion devices. In this network address, the first three integers are the *subnetwork identifiers*, while the last integer is the *host identifier*.

As the number of devices outpaced the number of available IPv4 addresses, there were not enough addresses to uniquely identify all devices. Therefore, a new addressing system was required which could uniquely identify more hosts. IPv6 is the IP version in which this became possible. In this version, entities are identified with 128 bits. One IPv6 address consists of eight groups of four hexadecimal digits. These groups are separated by colons. A hexadecimal is a positional system in which numbers are represented using base 16. Each sequence of four bits identify the numbers 0 until 9 and A until F. Thus, the first two groups of a IPv6 address 2001:0db8 is a representation of a sequence of eight times four bits: 0010 0000 0000 0001 0000 1101 1011 1000.

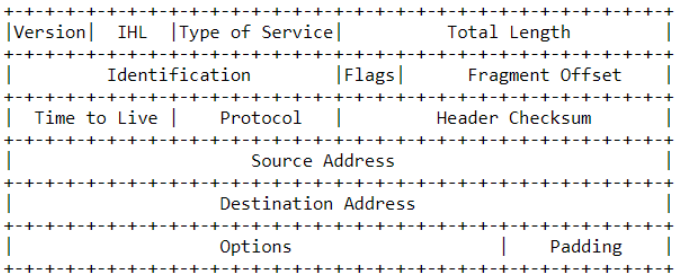


Figure 2.3: IPv4 packet header (Postel, 1981a)

Each IP packet consists of two parts: a packet header and its payload. Figure 2.3 shows the packet header of an IPv4 packet. The header of a packet is the metadata of the traffic being transferred over a network. These headers provide information concerning the communicating hosts which can be used as sensor data for the intrusion detector. Important headers are the *Source Address* and *Destination Address* which state the IP addresses of the communication hosts. The *Protocol* section indicates which transport layer protocol is used. In *Version*, the IP version in usage is stated, which is either 4 or 6. In the *Total Length* header section, the length of the IP packet (header and payload) is stated. The only difference between the packet headers of an IPv4 packet

and an IPv6 packet is the bit space of the addresses header.

2.3.2 Transport Layer

The two main protocols for sending data from one host to another in the transport layer are the User Datagram Protocol (UDP) and the Transmission Control Protocol (TCP). The difference between these protocols is that the former is connectionless, while the latter is connection-oriented.

User Datagram Protocol

UDP provides an unreliable connectionless transport service (Bonaventure et al., 2011). This protocol provides a way for applications to send data without the requirement of establishing a connection. Figure 2.4 shows the four headers of a UDP header. The *Source/Destination Port* identifies the ports used in the communication between two hosts. Each hosts has $65,535$ ports on which it can receive application specific information. With the use of ports, multiple applications can communicate at the same time. This is the main benefit of the UDP service over connectionless network layer service. The *Length* section shows the length of the UDP packet in bits. One application protocol which uses UDP is DNS, which will be discussed later. The *Checksum* section is a code used to protect data integrity.

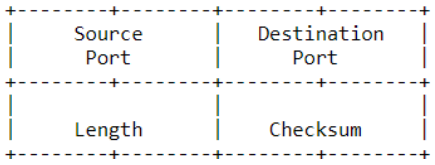


Figure 2.4: UDP header (Postel, 1980)

Transmission Control Protocol

In contrast to UDP, TCP provides a reliable connection-oriented transport service. This protocol is used by many applications because of the delivery guarantee. In this protocol, connections are established before connections can be used. Hosts confirm received packets by replying to each received packet with a packet containing an *ACK*(nowledgement) flag. To keep the order of the packets in place, each packet is identified with a *Sequence Number*.

A TCP connection is established with a *three-way handshake*. Figure 2.5 illustrates this handshake. Suppose host 1 wants to establish a TCP connection with host 2. The first action of host 1 is to send a packet containing a *SYN*(chronize) flag to host 2. The receiving host 2 will, in case it is willing to communicate, reply to this connection attempt with a *SYN* flag and an *ACK* flag by which it confirms that the *SYN* message from host 1 is received. When this acknowledgement is received by host 1, the connection is established and both hosts can send data to each other.

TCP connections can be terminated in two different ways. The first one is a graceful connection release, while the second is a more abrupt release. In the graceful way, the host will send a *FIN*(al) flag at the end of transmitting all data. This flag indicates that the final data packet has been sent and the host wants to terminate the connection. If the receiving host of this flag also wants to close the connection, it can send a FIN flag and the connection is closed. The abrupt connection release is simple and can be performed by one host sending a segment with the *RST*(reset) flag.

Figure 2.6 shows the section of a TCP header. Similar to UDP, TCP uses ports to allow multiple applications to send packets from source to destination. The *Sequence Number*, *Acknowledgment Number*, *Window* and *Checksum* are important for sending data in a reliable manner. This

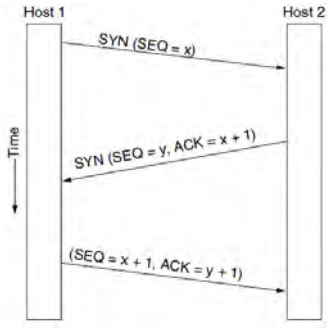


Figure 2.5: TCP three-way handshake (Tanenbaum & Wetherall, 2011)

information can be used to prevent congestion over the network. The performance of protocols is however not the scope of this research. In between the reserved part and the *Window* part are the flags which can be turned on or off. The *URG*(ent) and *PSH*(push) flag can be used to send missed data or send data that is urgent.

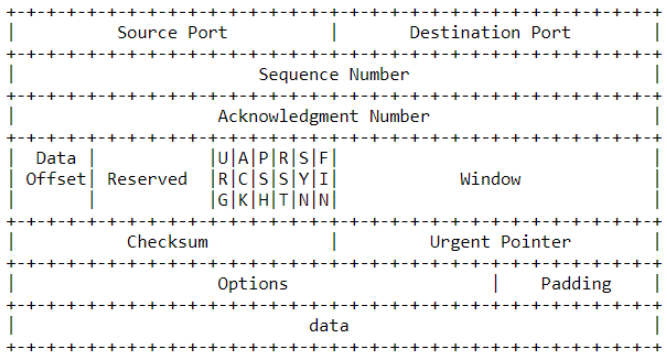


Figure 2.6: TCP header (Postel, 1981b)

2.3.3 Application Layer

On top of the layer stack in the reference model is the application layer. This is the most visible layer for users in computer networks in the sense that users can interact with applications in this layer. We will describe the protocols in this layer that are used in this research to detect intrusions. The protocols of our interest are: Domain Name System protocol, File Transfer Protocol, HyperText Transfer Protocol, Secucre Socket Layer and Secure Shell.

Domain Name System

In theory and in practice, resources could be reached by their IP addresses. But remembering these addresses by heart is not human-friendly. Furthermore, when (Web) servers or hosts change to a different IP address, all other hosts should be notified to this change. To overcome this issue, high-level, readable names were introduced to disconnect machine names from machine addresses. Automatically mapping these higher-level names with actual IP addresses is a service which is provided by the Domain Name System (DNS) protocol. The DNS itself is a hierarchical distributed database with many so called nameservers containing parts of the database.

Mockapetris (1987) recommends that all domain names should be structured according to the so-called Backus-Naur form. This is a technique to describe the syntax of a language, or in this case,

the syntax of domain names. The specified grammar states that a domain name must be read from left to right and sub-domains are separated by the dot character. For example, *www.nos.nl* corresponds to the domain *www* inside the domain *nos* that belongs to the *nl* top-level domain. Figure 2.7 illustrates the zones for which an nameserver provides the service of the DNS protocol. At the highest node of the hierarchy are the top-level domains and the root nameservers. These top-level domains can be recognized by the e.g. *.com*, *.org* and *.nl* suffix at the end of a uniform resource locator (URL), one higher-level naming scheme. Domain names are often included in a URL, but it is just one piece of the entire URL. The *resolver* then maps names into IP addresses.

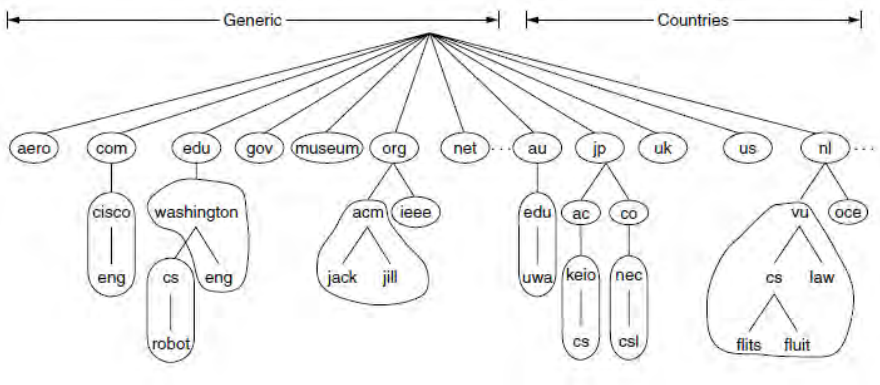
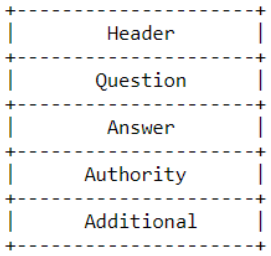


Figure 2.7: DNS nameserver zones (Tanenbaum & Wetherall, 2011)

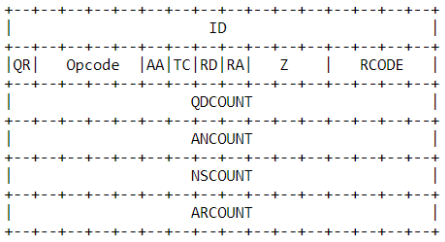
The DNS protocol consists of hosts sending DNS messages to these resolvers. A DNS message consists of five parts, which can be seen in Figure 2.8a. The message starts with a *Header*, followed by the *Question* for the nameserver or resolver. The *Answer* section is empty in the request, but will be used in the response message. In the *Authority* section, information can be provided by servers with a certain authority over some domain. In the last section, *Additional* information can be placed by the resolver or server that was not requested in the question.

Figure 2.8b shows the information stored in the DNS message *Header*. The *ID* is a random value chosen by the client. By this value, the server can store the question and it responds to the question. The *QR* flag is 0 for DNS queries and 1 in DNS answers. The *Opcode* is the type of query. *AA* is a bit that is set when the response has authority for the domain name. The *TC* bit specifies that this message was truncated due to length greater than permitted on the transmission channel. *RD* is set when the client requires the name server to pursue the query recursively. In the *RA* bit, the server responds whether this was supported or not. The *Z* bit is reserved for future usage. The *RCODE* is the response code of the server and has supported values between 0 and 5 according to Mockapetris (1987). The last four sections specify the number of entries in the question, answer, authority records and additional records section, respectively.

The header section of a DNS message is followed up by the query section. Figure 2.9a shows the format of this query section. *QNAME* is the domain name represented as a sequence of labels. The *QTYPE* is a code that specifies the type of the query, while *QCLASS* specifies the class of the query. Figure 2.9b shows the format of the Answer, Authority and Additional sections. In the resource records, the *NAME* section specifies the domain name to which this resource record pertains. The *TYPE* field defines the meaning of the data in the *RDATA* field, while *CLASS* specifies the class of this data. The *TTL* is a integer that determines the time interval in which the resource record may be cached. When this is 0, the resource record should not be cached. *RDLENGTH* defines the length of the *RDATA* field, which is a variable length string of bytes that describes the resource.

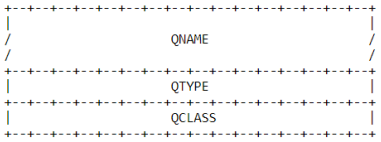


(a) Message format

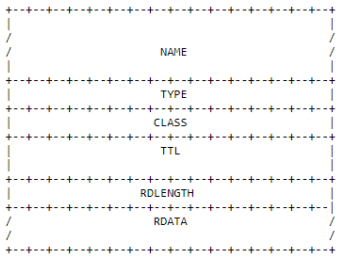


(b) Message header

Figure 2.8: DNS meta information message and header sections (Mockapetris, 1987)



(a) Query format



(b) Resource record format

Figure 2.9: DNS query and resource records format (Mockapetris, 1987)

File Transfer Protocol

The internet was in the beginning primarily used for remote terminal access with e-mail and file transfer. The File Transfer Protocol (FTP) is one of the protocols which provides the service of sending a file from a server to a client. In this context, a file can be an arbitrary object (document, spreadsheet, etc.). Sending a file across the internet is a complicated task, as computers are heterogeneous, in the sense that each computer system only supports some file representations, type information naming and access (Bonaventure et al., 2011).

In a FTP interaction, the client establishes a TCP connection to a FTP server and sends a series of requests to which the server responds. The server does not send data responses on the same connection as the client sends requests. Each time the server needs to download/upload a file, the server opens a new connection. The original connection is called a *control connection* over which commands are sent between the server and the client. The new connections that the server establishes with the client are the *data connections*. Before a client makes a request for a file, it first allocates a protocol port on the local operating system and sends this port number to the server to use as a connection port. In Figure 2.10, a schematic overview of the FTP interaction is given. It shows how server and user communicate using two different connections: one for data transfer and one for sending commands or receiving replies. In this figure, the PI is the protocol interpreter and the DTP is the data transfer process.

HyperText Transfer Protocol

At the heart of the World Wide Web is the HyperText Transfer Protocol (HTTP), an application-layer protocol that defines how Web clients request pages from Web servers and how servers transfer these pages to clients. HTTP consists of two programs: one for the client, who makes a requests, and one for the server, who responds to the request. The client side in HTTP is implemented in Web browsers, such as internet Explorer. HTTP typically utilizes TCP as underlying

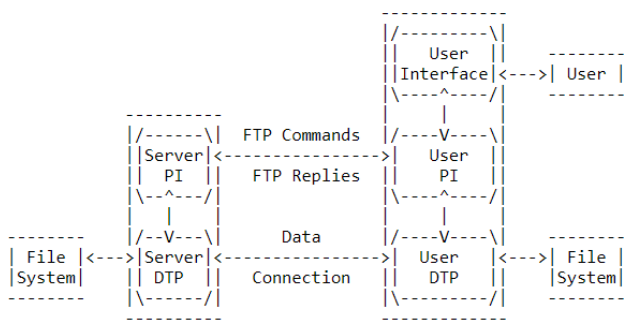


Figure 2.10: FTP model (Postel & Reynolds, 1985)

transport protocol.

The first documented HTTP version, version 0.9, was very limited in the sense that only one method was available for the client and the server could only send hypertexts. The HTTP protocol versions that are nowadays mainly used in practice are the HTTP 1.0 and HTTP 1.1 versions. These versions support more methods for clients and the response objects are not limited to simple hyper texts anymore. The difference between these versions is that the 1.1 has the option to keep TCP connections alive. In version 1.0, TCP connections are closed after the response of the server. The latest version, 2.0, aims at outperforming 1.1 by using ordered, bidirectional streams over one TCP connection (Grigorik, 2013). As 2.0 are not considered relevant for this research, only the semantics of the 1.0 and 1.1 versions are discussed.

In HTTP communication, clients and servers communicate by sending text messages. Figure 2.11 illustrates the format of both messages. In a request, the header starts with a request line, followed by request headers. In the response message, the header starts with a status line, followed by the response headers.

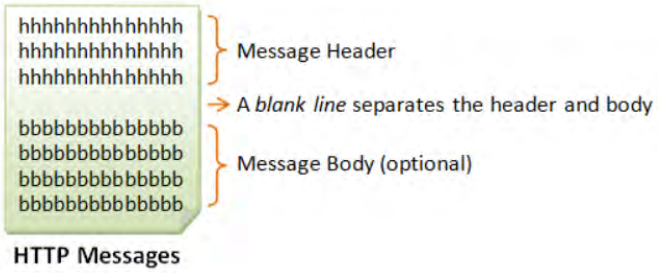


Figure 2.11: HTTP message format (Hock-Chuan, 2009)

A request message will start with a request line in the message header. This header has the following syntax:

method URI HTTP-version - *Request Line*

The request *method* defines what method the client wants to perform on the resource defined in the uniform resource identifier (*URI*). Table 2.3 gives a list of some methods and their descriptions. In the request message, the client will end the request line with a *HTTP-version* indicating which HTTP version the client wants to use.

Files can be reached by their *URI*, which is a unique string of characters. These URIs follow

Table 2.3: Subset of HTTP request methods (Tanenbaum & Wetherall, 2011)

Method	Description
GET	Read a Web page
HEAD	Read a Web page's header
POST	Append to a Web page
PUT	Store a Web page
DELETE	Remove the Web page
TRACE	Echo the incoming request
CONNECT	Connection through a proxy
OPTIONS	Query options for a page

a predefined set of syntax rules. The URI generic syntax consists of a hierarchical sequence of components referred as the scheme, authority, path, query and fragment (Berners-Lee, Fielding, & Masinter, 2005). A URI example is given in Figure 2.12 with all components given. These components are delimited by a set of special characters, namely: $?, :, \#, /, [,], @, !, \$, \&$. We call these character "reserved" as their usage is limited to the purpose of separating components.

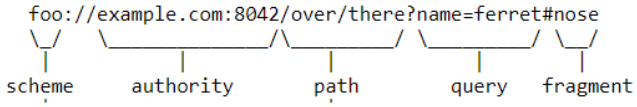


Figure 2.12: URI example (Berners-Lee et al., 2005)

In the status line of a response message, the HTTP version is given, a status code and a reason phrase:

HTTP-version status-code reason-phrase - *Status Line*

The *HTTP-version* is the version in which the server responds and accepts. The *status-code* is a 3-digit number that reflects the outcome of the request. The status-codes can be grouped by their first digit, as can be seen in Table 2.4. The *reason-phrase* gives an explanation of the status code.

Table 2.4: HTTP status codes (Tanenbaum & Wetherall, 2011)

Status code	Description	Example
1xx	Informational	100: server agrees to handle client request
2xx	Successful	200: request succeeded
3xx	Redirection	301: page moved
4xx	Client error	404: page not found
5xx	Server error	500: internal server error

2.4 Security in Network Communication

Suppose two hosts want to communicate over the internet without intruders sniffing in their data. One solution to prevent sniffing is by encrypting the transferred data. Hosts can encrypt sensitive data in such way that sniffers cannot understand the messages sent. Before encrypted data can be sent, both hosts must share keys which can be used to decrypt the message. With this technique, data can remain confidential. The Secure Socket Layer and the Secure Shell both provide the service of securing communication.

2.4.1 Transport Layer Security/ Secure Socket Layer

On top of the TCP transport layer is the Secure Socket Layer (SSL), which provides the service of encrypting TCP connections. The SSL is the predecessor of the Transport Layer Security (TLS). This service is often used to secure information that is sent over the internet, such as transactions. In practice, HTTPS is basically HTTP combined with SSL security. SSL data transmission are often sent on port 443. In the establishment of a TCP connection with SSL protection between host A and host B, four keys are generated with cryptographic algorithms to encrypt and decrypt the data:

- E_A = session encryption key for data sent from A to B
- M_A = session MAC key for data sent from A to B
- E_B = session encryption key for data sent from B to A
- M_B = session MAC key for data sent from B to A

The Message Authentication Code (MAC) key is used to keep the data integrity intact, while the encryption key is used to keep the confidentiality of the data intact. When data is sent from host B to A, Figure 2.13 shows that the data is encrypted as well as the MAC. In front of the data are the record *Type*, *Version* of the SSL protocol in use and *Length* of the packet. Available SSL versions are 1.0, 2.0 and 3.0.

Figure 2.13: SSL record (Kurose & Ross, 2012)

2.4.2 Secure Shell

Secure Shell (SSH) is the predecessor of Telnet and provides the service of operating network services over a secure line. Applications often used here are remote command-login, login and remote command execution. SSH connections mainly run over port 22. SSH is similar to SSL as they both use the same cryptographic algorithms. The difference is that SSL uses digital certificates while SSH does not.

Chapter 3

Intrusion Detection Systems

An Intrusion Detection System (IDS) is the "*burglar alarm*" in the field of computer security (Axelsson, 2000). One of the first IDS was created by Denning (1987). Since then, many IDS have been proposed in literature. To put the developed methods in perspective, This chapter provides the description of different IDS technologies and methods. The advantages and disadvantages of each of those concepts are given.

3.1 Intrusion Detection/Prevention Systems

Scarfone and Mell (2007) define the task of intrusion detection as *" ... the process of monitoring the events occurring in a computer system or network and analyzing them for signs of possible incidents, which are violations or imminent threats of violation of computer security policies, acceptable use policies, or standard security practices"*. An IDS is software that automates the process of intrusion detection on a computer or network system. The difference between an IDS and an Intrusion Prevention System (IPS) is that an IPS is assigned with both the task of detecting intrusions and preventing them. The IDS can therefore be seen as part of an IPS. Since we are interested in the crucial step of detecting intrusions, we will focus on the capabilities of IDS systems.

3.2 Detection Technologies

The two main IDS technologies are host-based and network-based intrusion detection (Axelsson, 2000). In this research we develop a network-based IDS, but it is still relevant to compare the host-based IDS with network-based variant IDS and name the advantages and disadvantages of both types of technologies.

3.2.1 Host-Based Intrusion Detection

A host-based Intrusion Detection System (HIDS) is integrated in a single host and looks for suspicious events in that machine. The detection software installed on the host is called an *agent*. In practice, these agents are often deployed at critical hosts such as publicly accessible servers and servers containing sensitive information. Figure 3.1 illustrates a HIDS deployment architecture. Here, IDS systems are labelled as IDPS, as the authors combined IDS and IPS into IDPS. Hosts have an IDS or IPS agent installed on its operating system. Other typical devices that can be found in a network, such as a router, a switch and a firewall, are also shown in this example. These devices are not part of the IDS, but it can be observed how the system is integrated in a network consisting of all standard components.

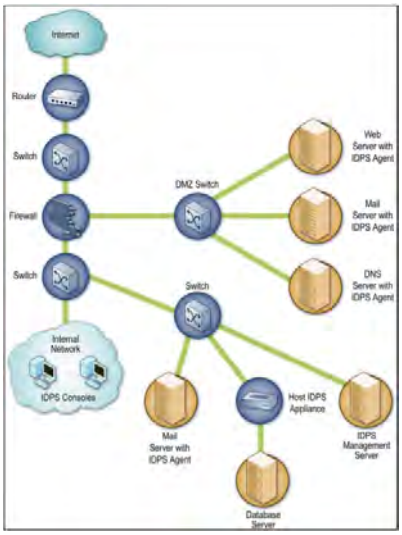


Figure 3.1: Host-based detection architecture (Scarfone & Mell, 2007)

One advantage of having an HIDS is that it can analyse the activities within encrypted network communication as it is the endpoint of the communication. As the HIDS is installed on the host and can obtain the keys of the corresponding host, it can decrypt messages and analyse these packets. The decrypted information is then used as input for the intrusion detector. Furthermore, as the HIDS is able to identify the host characteristics and configurations, deviating behaviour can better be detected. Scarfone and Mell (2007) identify the following drawbacks in HIDS:

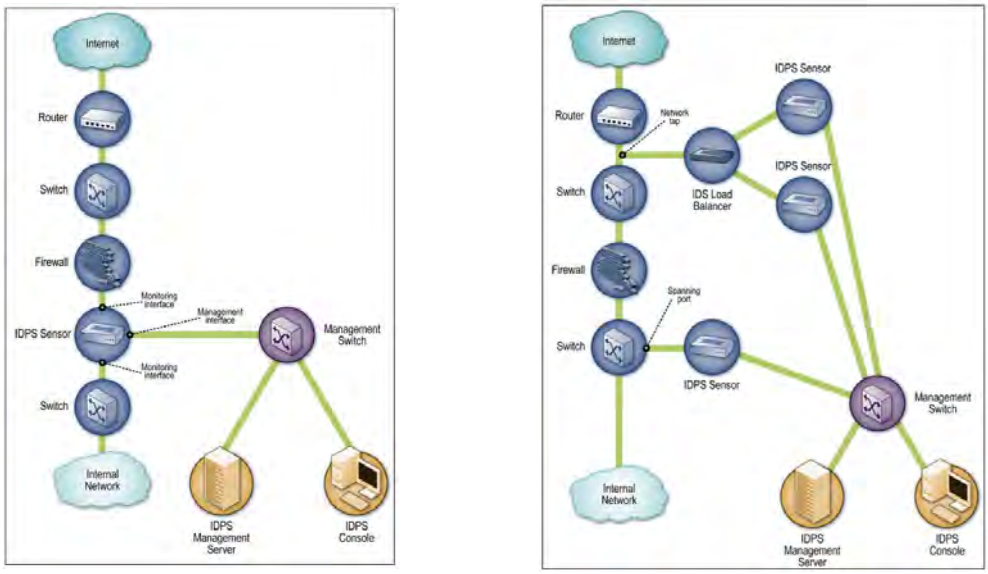
- As each host has an HIDS installed, considerable hosts resources are used, such as memory processor and disk storage.
- Identification of intrusions are not performed real-time as alert data is sent in batches to management servers to prevent overhead in the network.
- HIDS can be in conflict with existing security controls on the host.

3.2.2 Network-Based Intrusion Detection

A network-based Intrusion Detection System (NIDS) monitors network traffic for particular network segments or devices to identify suspicious activity on the network, transport and application protocol level. These systems are often implemented at the edge between networks. The sensors of a network-based IDS are either deployed inline or passive. Figure 3.2 gives an example architecture of both deployment techniques. An inline sensor is deployed in such way that network traffic is monitored by passing it through the sensor. These sensors are typically placed as network firewalls or other network security devices. In contrast, a passive sensor is placed in such way that the real internet traffic is copied and this copied traffic is used by the IDS as input data.

A NIDS can collect all sorts of data regarding hosts characteristics such as the operating system and the application usages. Furthermore, network characteristics can be taken into account. There are, however, some limitations for these kinds of systems (Scarfone & Mell, 2007):

- The method cannot detect attacks within encrypted network traffic by packet analysis.
- When the network is in high load conditions, full packet analysis cannot be executed due to overhead.
- Network-based detectors/sensors are vulnerable systems as they are susceptible to various types of attacks, mostly involving large volumes of traffic.



(a) Inline placement

(b) Passive placement

Figure 3.2: Network-Based IDS sensor architecture (Scarfone & Mell, 2007)

3.3 Detection Methodologies

This section describes the IDS methodologies. The major methods are the signature-based detection and anomaly detection methods. Another method which is not studied often is the stateful protocol analysis. As the stateful protocol analysis approach is similar to the approach used in this research, it is relevant to describe. This section will give a description of all three methodologies and state an overview of several advantages and disadvantages of the three methods.

3.3.1 Signature-Based Intrusion Detection

When known threats are identifiable by a set of patterns, these patterns form the *signature* of this attack. These signatures can be used to stop threats which comply to these patterns. One example of a signature is an attacker who tries to log in with a default username such as "admin" or "root" and a default password. In signature-based detection, also known as misuse detection or knowledge-based detection, observed events are compared with a list of these so-called signatures.

Figure 3.3 gives the scheme describing how intrusions are detected using a signature-based method. When events resemble a signature, the detection method gives an alert. It is extremely effective in detecting known threats. The drawbacks of this method are (Liao, Richard Lin, Lin, & Tung, 2013):

- It lacks effectiveness in detecting previously unknown threats and variants of known attacks.
- It is time consuming to maintain knowledge and keeping the signature list up to date (Debar, Dacier, & Wespi, 1999).
- It does not track and understand complex communication because it only looks at information "on the fly" without looking at previous information. The system is in some sense memoryless as it only tries to identify patterns which could indicate malicious events.

Signature-based intrusion detection systems are often combined with honeypots. A honeypot is a defensive tool to trap attackers by fooling the attacker thinking they are attacking a real system while it is actually a decoy. It is possible to find new attacks and gather new rules to automatically

detect some of the new attacks (Yon Tang & Shigang Chen, 2005). One issue with honeypots is that when attackers know that a network has honeypots, they will avoid these honeypots in next attacks.

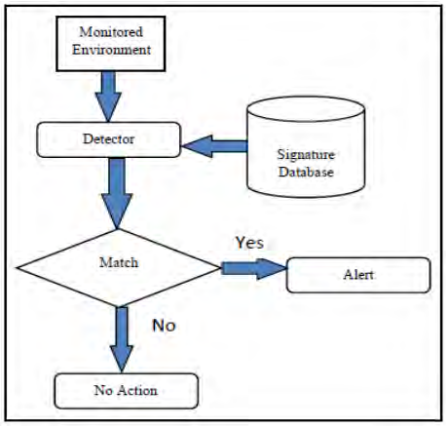


Figure 3.3: Signature-based detection architecture (Mudzingwa & Agrawal, 2012)

3.3.2 Anomaly-Based Intrusion Detection

In anomaly-based detection, also known as behaviour-based detection, events are compared with each other to identify which instances are significantly different. One major assumption in anomaly-based detection is that intrusions behave differently than normal behaviour. These intrusions are considered outliers or anomalies. Figure 3.4 illustrates the architecture of anomaly-based detection. The method learns profiles which represent activities generated by normal users on a network. These profiles are developed by monitoring network characteristics over a period of time. Activities that differ from the expected behaviour might indicate that they are malicious.

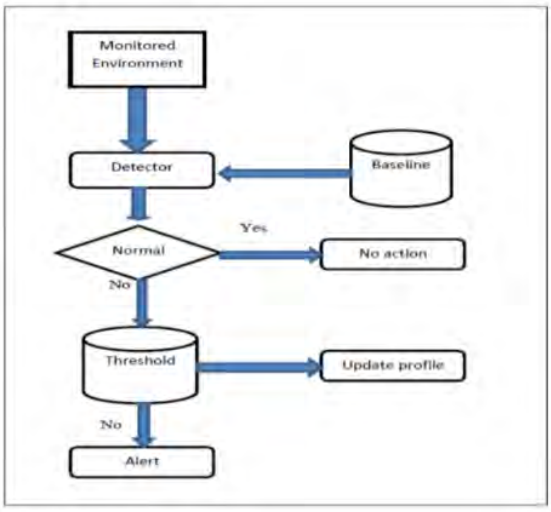


Figure 3.4: Anomaly-based detection architecture (Mudzingwa & Agrawal, 2012)

One major advantage of this method is the ability to identify previously unknown attacks. Profiles are generated by training over a period of time. These profiles can be either static, but can also be dynamic over time. As the system changes over time, the corresponding profiles should change over time. The drawbacks of this methods are (Scarfone & Mell, 2007):

- Creating profiles which represent normal behaviour is challenging by the complexity of reality. User behaviour is very diverse so constructing profiles can be difficult.
- Alerts are hard to validate as the number of events causing the alert is hard to analyse by the complexity of the model or the events.

3.3.3 Stateful Protocol Analysis

A third method in Intrusion Detection Systems is stateful protocol analysis, or "deep packet inspection". Stateful protocol analysis is the process of comparing predefined profiles of generally accepted definitions of benign protocol activity against observed events on the network. In contrast to the previous methods, stateful protocol analysis relies on vendor-developed profiles that determine acceptable and unacceptable protocol events. The term 'stateful' indicates that the IDS should track the state of all protocol levels (network, transport, application). For example, for a TCP-connection to be established, the connection starts in an unestablished state. If the connection is established with the normal three-way handshake, then the state changes to an established connection state. When the three-way handshake is not performed correctly, this is observed by the method. By monitoring commands, unexpected sequences of commands can be detected.

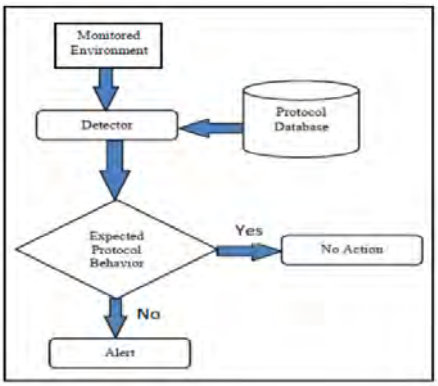


Figure 3.5: Stateful Protocol Analysis Architecture (Mudzingwa & Agrawal, 2012)

Figure 3.5 shows the general architecture of the stateful protocol analysis. Behaviour is compared with the expected behaviour observed in the protocol. If the behaviour deviates from the accepted behaviour, the IDS alerts the user. There are, however, some drawbacks with this method (Scarfone & Mell, 2007):

- The analysis is very resource-intensive by state tracking in many sessions and the complexity of the task.
- The method cannot detect attacks that do not violate the standards of the accepted protocol behavior.
- The protocol model used by the IDS might conflict with implemented protocols.

As stateful protocol analysis did not succeed to be integrated and adapted in IDSs because of the discussed drawbacks, most research mainly concentrates on anomaly-based and signature-based methodologies (Mudzingwa & Agrawal, 2012). In this research, protocol specified information is taken into account for the construction the anomaly detector. However, we do not take the state of the protocol in consideration but we use features describing if the protocols are actually vendor-defined utilized. Therefore, the constructed features utilizes the idea of stateful protocol analysis. To be more precise, Chapter 6 will show which features are created for the different protocols.

Chapter 4

Literature Review

Anomaly or outlier detection is the study of finding patterns in data which do not conform to expected or normal behaviour. The first work which combined multiple outlier detection techniques from different fields is given in Chandola, Banerjee, and Kumar (2007). Later on, the same authors Chandola, Banerjee, and Kumar (2009) provide an overview of anomaly detection techniques in multiple application. Applications in which anomaly detection is applied are intrusion detection, fraud detection, fault detection, medical informatics, image preprocessing and sensor networks. The authors define six techniques in which anomaly detection can be applied (e.g. classification, clustering, and nearest neighbour based). Nowadays, anomaly detection algorithms are combined with Deep Learning techniques, such as auto encoders and convolutional neural networks (Chalapathy & Chawla, 2019). Furthermore, real-time anomaly detection is of interest in the literature, especially for big data datasets (Habeeb et al., 2019).

One of the most recent anomaly detection application in cybersecurity is given in Klein, Bhulai, Hoogendoorn, Van Der Mei, and Hinfelaar (2018). Unsupervised techniques are applied on a partially labelled dataset. As the data is partially labelled, it is hard to measure whether the results are valid. This dataset is however not publicly available in contrast to the datasets that are used in this research. Our research has however a lot of similarities in the approach manner.

Akoglu et al. (2015) provide a general, comprehensive and structured overview of the state-of-the-art methods for anomaly detection in data represented as graphs. The authors distinguish four types of anomalous behaviour which can be measured in graphs: anomalous vertices, edges subgraphs and/or event and change detection. In the last part, change in the dynamic changing network is detected, which is also of interest in this research. To detect these anomalous types, Akoglu et al. (2015) determine five methods: community detection, compression, matrix/tensor decomposition, distance metrics and probabilistic models. In this research, the focus is on finding suitable distance metrics to identify malicious network changes.

Pincombe (2005) uses ten different distance metrics on TCP/IP traffic to identify differences between sequential networks. These distances are used as time series to fit an ARMA model. This model makes a prediction of the value for the next distance. When the calculated residual, which is the difference between the expected and the real value exceeds an certain threshold, the change is flagged as anomaly. Gaston, Kraetzl, and Wallis (2006) present the idea of detecting abnormal change in the network time series by using the graph diameter as distance metric between sequential graphs. Papadimitriou, Dasdan, and Garcia-Molina (2010) propose five similarity schemes to determine the similarity between two graphs. The Signature Similarity scheme is a very good measure for detecting skipped rows, missing connected sub-graphs and missing random vertices. Berlingerio, Koutra, Eliassi-Rad, and Faloutsos (2012) propose the NetSimile algorithm which calculates similarities by extracting structural features for each graph and comparing these features between graphs by the Canberra Distance. The structural features are consisting of node

information such as the number of neighbours, clustering coefficients and information regarding Egonets (Akoglu, McGlohon, & Faloutsos, 2010). Koutra, Vogelstein, and Faloutsos (2013) state that their algorithm DeltaCon is a principled, intuitive, and scalable. The Fast Belief Propogation is used to compute node affinities, where most other studies use Pagerank, personalized Random Walks with Restarts (RWR) or lazy RWR. The pairwise node affinity scores are computed for each graph and the differences between them are calculated using the Matusita distance.

Le, Jeong, Roman, and Hong (2011) used traffic dispersion graphs to model network traffic over time. There are two metrics to describe the graphs: static metrics and dynamic metrics. To calculate the distance between two consecutive graphs, the joint degree distribution is calculated for both graphs and afterwards the euclidean distances is taken between these JDD's. This method is validated using the POSTECH and CAIDA dataset containing DDoS attacks.

Chapter 5

Data

This chapter discusses the three datasets used in this research. First, the challenges in creating intrusion detection datasets are given. These challenges give an intuition why the perfect intrusion detection dataset does not yet exist. Researchers have tried to generate various intrusion detection datasets and three of them were selected for this research: the ISCX-IDS-2012, CIC-IDS-2017 and the UNSW-NB15 dataset. A detailed description is given on how the datasets were composed. For the ISCX-IDS-2012 and the CIC-IDS-2017 dataset, some inherent issues will be stated. At last, the data extraction will show how raw internet traffic is transformed into protocol specific log files.

5.1 Intrusion Detection Datasets

A major challenge in anomaly-based intrusion detection is the lack of appropriate publicly available datasets for evaluating and comparing intrusion detection systems. This gap is caused by the nature of the data: inspecting network traffic can expose sensitive information, including confidential or personal communication (Sommer & Paxson, 2010). Besides that, publicly sharing raw internet traffic might violate privacy laws. To overcome this problem, researchers have studied two possible solutions: *simulation* and *anonymization*. The advantage of using simulation a network traffic generator is that it is free of privacy or sensitivity concerns. Unfortunately, as real internet traffic is very complex, simulating realistic traffic is extremely difficult (Floyd & Paxson, 2001). Anonymizing internet traffic has gained little traction as fear exists that sensitive information can still leak, as can be seen in (Coull, Wright, Monrose, Collins, & Reiter, 2007).

Most intrusion detection systems are tested on two public available datasets: DARPA and the KDD Cup dataset derived from the DARPA dataset (Sommer & Paxson, 2010). These datasets are, however, two decades old and are not adequate for benchmarking current practice. This is because network behaviour has evolved and attacks are different than twenty years ago. Therefore, the community responded by creating multiple new datasets in the last years. Ring, Wunderlich, Scheuring, Landes, and Hotho (2019) provide an overview of all network-based intrusion detection datasets from 1998 until 2017. In total, 34 different datasets were investigated and a short description for each dataset is given. In the view of the authors, a perfect network-based dataset should i) have up-to-date and a broad range of attacks, ii) have correctly labelled traffic, iii) be publicly available, iv) contain real network traffic and v) span over a long period.

To ensure reproducibility, intrusion detection methods should be tested with at least one publicly available dataset. It is even better to use several public datasets to avoid overfitting to a certain dataset. By recommendation of Ring et al. (2019), the selected datasets were the ISCX-IDS-2012 (Shiravi, Shiravi, Tavallaee, & Ghorbani, 2012), the CIC-IDS-2017 (Bhuyan, Bhattacharyya, & Kalita, 2015) and the UNSW-NB15 (Moustafa & Slay, 2015) dataset for IDS evaluation. The next

sections describe how the data is generated and how normal and malicious traffic is generated for all datasets.

5.2 ISCX-IDS-2012

Shiravi et al. (2012) suggest to move away from *static* and *one-time* intrusion detection datasets, because internet networks are not stationary but dynamically evolving. Instead, the authors suggest that datasets should be dynamically generated with up-to-date intrusions and latest network behaviour. These datasets should be modifiable, extensible and reproducible to ensure that the network traffic composition reflects recent network developments. In this view, the authors composed a systematic approach to generate such datasets by simulating network traffic to generate frequently new datasets rather than one-time dataset per study. This traffic is generated by using the notion of so-called *profiles*. A profile contains an abstract representation of network events and behaviour. To be more precise, internet events generated by real users are abstracted in profiles. These profiles are used by computer agents to mimic user's behaviour to simulate internet traffic. By using the abstract version of user's behaviour, the behaviour does not depend on the structure of the network. The dataset which is generated using this approach by the authors is the ISCX-IDS-2012.

5.2.1 Experimental Design

The ISCX-IDS-2012 dataset was generated by continuously simulating internet traffic for exactly one week starting from Friday 11 June until Friday 18 June 2010. internet traffic is generated by using two types of profiles to compose benign and malicious traffic. On the one hand, attack profiles attempt to represent attack scenarios, while on the other hand the benign profiles are used to generate benign internet activities. There are four attack scenarios constructed, for which different attacks are used. The scenarios and attacks can be found in Table 5.1.

Table 5.1: ISCX-IDS-2012 attack scenarios

Scenario	**Day**	**Description**	**Attacks**
1	Sunday 13 June	Infiltrating the network from the inside	Infiltrations, backdoors, portscans, exploits
2	Monday 14 June	HTTP denial of service attack	Portscans, exploits, DDoS slowloris
3	Tuesday 15 June	DDoS using an IRC Botnet	Backdoor, DDoS Botnet
4	Thursday 17 June	Brute Force SSH	SSH Brute Force, exfiltration

The β-profiles are composed by abstracting internet behaviour of the Canadian Institute for Cybersecurity (CIC) users. The following protocols were used to profile user's activity: HTTP, FTP, SSH and e-mail protocols. The DNS protocol is indirectly used for mapping hosts with their corresponding IP addresses. Figure 5.1 shows the testbed architecture of the experiment. 21 interconnected workstations are divided over four distinct sub-networks. The fifth LAN consists of three servers that provide web, email, DNS and other services. A sixth LAN is set up to conduct monitoring and maintenance of workstations and servers. A set of servers is used for monitoring and capturing the network traffic. Hosts have multiple IP addresses (IPv4 private, IPv4 public and IPv6). A mapping of these addresses and the corresponding host can be found in Appendix A.1. This address mapping can be used to map network traffic to the designated host.

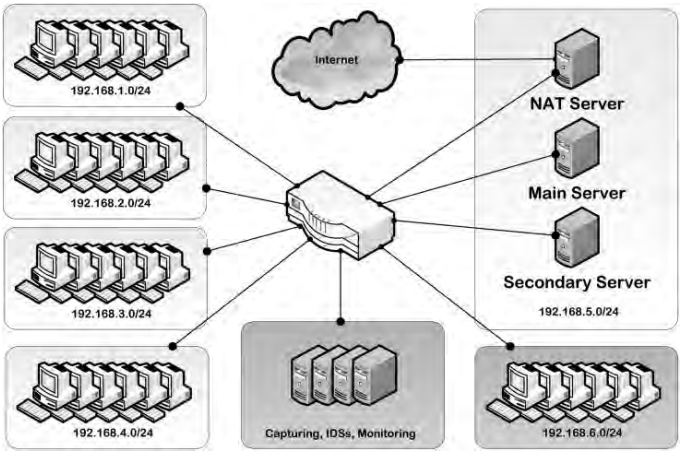


Figure 5.1: ISCX-IDS-2012 testbed network architecture (Shiravi et al., 2012)

5.2.2 Inherent Problems

Unfortunately, there are some issues with the ISCX-IDS-2012 data. First of all, the dataset provided by the authors is not directly usable for applying machine learning methods. With several categorical features, some machine learning algorithms cannot directly be applied. This implies that this dataset cannot be used as a benchmark for all machine-learning techniques. Secondly, the authors did not provide a labelling scheme to manually check the connections. In the dataset, connections are either benign or malicious, so it is not verifiable which attack is occurring when connections are malicious. On the basis of the attacks descriptions in Shiravi et al. (2012), a new labelling scheme was created for the ISCX-IDS-2012 dataset, which can be found in Appendix A.2. The original dataset provided by the authors was used to acquire and to confirm the time stamps, hosts (IP addresses and ports) and attacks.

When comparing the meta-data of the attacks in the paper and the meta-data of the attacks in the dataset, there appears a striking difference. Some connections which are labelled as malicious are not mentioned in the paper. On Sunday 13 June for example, 314 connections are labelled as attacks, but are not acknowledged in the paper. Other malicious classified connections which are not discussed in Shiravi et al. (2012) can be found in Table 5.2. The $*$ symbol indicates that multiple ports are used.

Table 5.2: ISCX-IDS-2012 connections labelled as malicious without documentation

Start	End	Orig Address	Resp Address	Orig Port	Resp Port
14-6 18:26	14-6 21:26	192.168.3.117	131.202.241.200	2070	80
14-6 21:17	14-6 21:17	217.140.88.36	192.168.5.122	17316	22
14-6 22:43	14-6 22:44	200.153.221.172	192.168.5.122	46216	22
15-6 02:58	15-6 02:58	212.77.187.249	192.168.5.122	*	22
15-6 21:36	15-6 23:38	217.76.44.243	192.168.5.122	*	22
15-6 11:16	15-6 11:16	141.64.8.215	192.168.5.122	*	22
17-6 15:09	17-6 15:13	58.211.72.43	192.168.5.122	*	22
17-6 16:32	17-6 16:36	131.202.243.90	192.168.5.122	*	21
17-6 21:35	17-6 22:00	131.202.243.90	192.168.5.122	34446	22

5.3 CIC-IDS-2017

The CIC-IDS-2017 dataset is one of the most recent generated network intrusion detection datasets. Sharafaldin, Habibi Lashkari, and Ghorbani (2018) state that the dataset covers all necessary criteria which are required for a network-based intrusion detection dataset. One such criterion is that network intrusion detection datasets should contain up-to-date attacks. The authors use a wide variety of popular attacks, such as denial of service attacks and port scans. The provided dataset was constructed using CICFlowMeter, a Java developed program by the authors, that derives bidirectional flows from raw internet traffic into 80 statistical features. The authors prioritize generating realistic background traffic when creating an intrusion detection dataset. To this end, the authors also use benign profiles similar to Shiravi et al. (2012). This system profiles the abstract behaviour of human interactions and generates realistic normal background traffic by using these profiles. In the CIC-IDS-2017 dataset, 25 different B-Profiles are used based on HTTP, HTTPS, FTP, SSH and email protocols.

5.3.1 Experimental Design

The capturing period of the network traffic consists of five work days. The experiment started on Monday 3 July 2017 and ended on Friday 7 July 2017. Raw internet traffic was daily captured from 9 a.m. until 5 p.m. The first day of the experiment only contains normal traffic, while the other days of the experiment contain both benign and malicious traffic. The attacks can be found in Table 5.3.

Table 5.3: CIC-IDS-2017 attack scheme

Day	**Attacks**
Tuesday 4 July	Patator - FTP, Patator - SSH
Wednesday 5 July	DoS Slowloris, DoS SlowHTTPTest, DoS Hulk, DoS Goldeneye, Heartbleed
Thursday 6 July	Web Attack Brute Force, Web Attack XSS, Web Attack SQL Injection, Infiltrations, Portscans
Friday 7 July	Botnet Ares, Port Scans, DDoS LOIC

The testbed architecture is divided into two networks. Figure 5.2 shows these two networks. On the left side, there is the victim network, which consists of all usual components of a normal network. Similar to the ISCX-IDS-2012, there are several servers which provide services to the workstations in the victim network. On the right side, there is an attack network with four operating systems. These are the workstations from which attacks were executed. Sharafaldin et al. (2018) only provide IPv4 addresses of the testbed architecture, but we gathered additional information about the victim network addresses such as the IPv6 and the MAC addresses. This information can be used to link IP addresses with their corresponding host machine. These associations can be found in Appendix B.1.

5.3.2 Inherent Problems

The CIC-IDS-2017 dataset is the successor of the ISCX-IDS-2012 dataset. The authors provide more meta-data about the experiment and have improved some issues of the ISCX-IDS-2012 dataset. While some problems were solved, there are still some issues with the new dataset. These issues can be divided in two parts: *feature problems* and *labelling issues*.

The CIC-IDS-2017 dataset was constructed by generating bidirectional flows using the CICFlowMeter[1] tool on the captured raw network traffic. The program is coded in such way that TCP connec-

[1]The CICFlowMeter version was downloaded in May 2019.

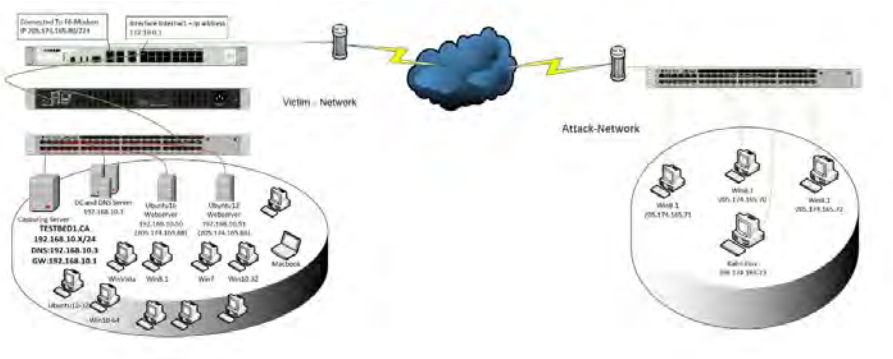


Figure 5.2: CIC-IDS-2017 testbed network architecture (Sharafaldin et al., 2018)

tions are closed when a packet with a FIN flag is received. Figure 5.3 shows the code comments in the CICFlowMeter which indicate TCP connections are stored after the first FIN flag. However, TCP communication does not necessarily stop at the first FIN flag. The receiver of the data can still send packets to acknowledge that the packet with the FIN flag has arrived. CICFlowMeter unintentionally splits TCP communication in two flows, where the last part could only consists of the receiver acknowledging that the FIN flag has arrived. This results in many TCP connections divided in two bidirectional flows and thus many redundant data instances.

```
// Flow finished due FIN flag (tcp only):
// 1.- we add the packet-in-process to the flow (it is the last packet)
// 2.- we move the flow to finished flow list
// 3.- we eliminate the flow from the current flow list
```

Figure 5.3: CICFlowMeter comments on ending TCP connections (Lashkari, 2019)

The creators of the CICFlowMeter state that 83 statistical features are generated, but not all of these features are relevant. For example, some features do not differ from their default value zero. In the code of the CICFlowMeter, it appears that these features were not updated when executing the program. Furthermore, there seems to be a mistake in flag counts in TCP segments. Only the flags in the first packet of a flow are counted. Flags in later packets were ignored for unknown reason to us. At last, some of the features are exactly same, but have different descriptions, such as *total_fpackets* and *total FWwd Packet* which both count the number of forwarded packets. This results in strong correlation in the independent variables, which is undesirable when applying anomaly detection algorithms.

To check the labelling scheme of the CIC-IDS-2017 dataset, the provided meta-data[2] of the attacks were compared with the meta-data of the attacks in the provided dataset. Apparently, there are some inconsistencies with the labelling scheme as well as some of the labels generated for the connections. There are some cases in which connections were labelled as malicious while they are actually benign and vice versa. Another issue in the published CIC-IDS-2017 is that the connections are only labelled in one direction. Connections from the attacker were labelled as malicious, but the connections from the victim to the attacker are labelled benign. Furthermore, the start times and end times of the attacks seem inaccurate as some of the attacks actually start earlier or later than stated in the scheme. Table 5.4 gives an overview of connections which seem to be labelled incorrectly labelled. To overcome these labelling issues, we provide an improved labelling scheme, which can be found in Appendix B.2.[3]

[2] https://www.unb.ca/cic/datasets/ids-2017.html

[3] In correspondence with the authors, they excluded attacks as it would "... affect results on the learning system".

Table 5.4: CIC-IDS-2017 list of discovered labelling issues

Day	Problem Description
Tuesday	Observation: One connection using port 80 is labelled as FTP attack Remark: FTP attack is on port 21 Amendment label: BENIGN
Wednesday	Observation: Several connections around 14:24 have the label DDoS Slowloris Remark: DDoS Slowloris was in the morning Amendment label: BENIGN
Thursday	Observation: The infiltration attack on network host 25 is labelled BENIGN Amendment label: Infiltration Observation: Portscans from host 192.168.10.8 to other hosts are BENIGN Amendment label: Portscan
Friday	Observation: Connections from 6.6.13.28 and 57.7.235.158 are Botnet attacks Remark: Botnet attack do not come from these IP addresses Amendment label: BENIGN

5.4 UNSW-NB15

The last dataset used for this research is UNSW-NB15 (Moustafa & Slay, 2015). This dataset is a combination of real traffic and attack activities in the network traffic. Here, normal traffic is generated using the IXIA PerfectStorm tool to create both kinds of traffic. Rather than creating profiles of several users, the authors use the IXIA tools to simulate a flow of normal traffic. The dataset was created by capturing two days: 16 hours on 22 January 2015 and 15 hours on 17 February 2015. In these days, both normal and malicious traffic traces were generated by the PerfectStorm tool. The following attacks were constantly executed during the simulation: Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode and Worms. On the first day of the simulation, one attack per second was generated, while on the second day of the experiment 10 attacks per second were executed. Figure 5.4 shows the testbed of the experiment. Three virtual servers were used to generate traffic. Servers 1 and 3 were configured to generate normal behaviour, while server 2 generated normal and malicious activities.

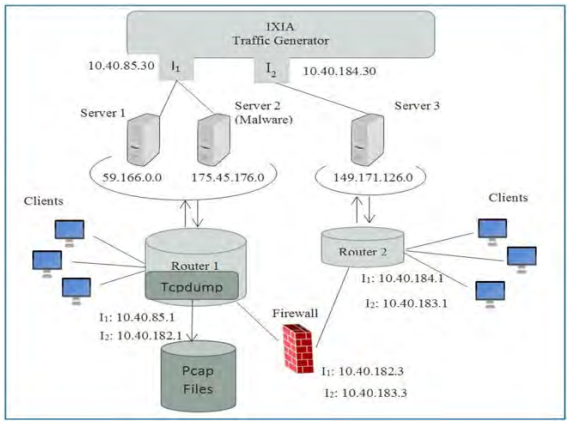


Figure 5.4: UNSW-NB15 testbed network architecture (Moustafa & Slay, 2015)

It seems rather optimistic to exclude attack so that the performance of learning systems increases as only obvious attacks will be learned.

5.5 Data Extraction

We acquired the raw internet traffic in .pcap or .pcapng form from each dataset. The latter format is the new generation (ng) pcap form. There were some broken packets in the raw internet traffic, so the tool pcapfix (version 1.1.4) is used to fix these packets. Afterwards, Zeek[4] (formerly BRO) is executed on the pcap files to generate log files for each specific protocol. Zeek is an open-source network traffic analyser which stores protocol specific information in separate files. Packets are read in an event engine and protocol specific log files are created where the events are stored in. For this research, the following log files and their according protocols are considered: conn.log (TCP,UDP), dns.log, ftp.log, ssh.log, ssl.log and http.log.

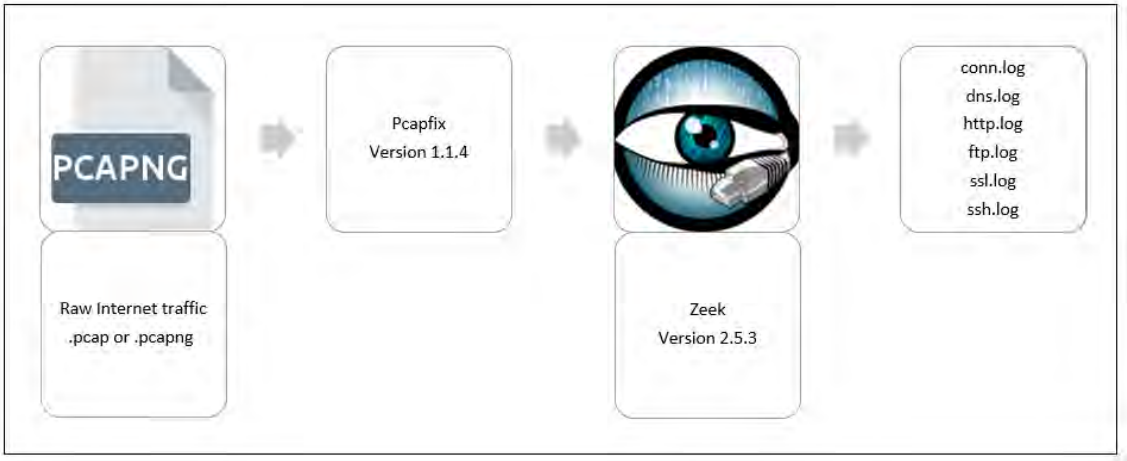


Figure 5.5: Data extraction steps

[4] https://www.zeek.org/

Chapter 6

Feature Engineering

The network traffic analyser Zeek is used to transform raw network traffic into log files. Each of these files describe the usage of a certain protocol observed on the network. For example, the *http.log* file consists of records which shows the HTTP header fields used in a session. This chapter describes how these protocol log files are prepared for machine learning purposes. First, the session features are discussed which uniquely identify each connection. Afterwards, for each of the considered protocols, it is shown how features are composed.

6.1 Connection Identification

Packets streams are divided in connections by the event engine of Zeek. Table 6.1 shows the identification features describing a single connection. Each connection is assigned with a random unique identifier to distinguish connections from another. The *id* features show the IP addresses and ports of the communicating hosts. The originator (orig) of the connection is the host who sends the first packet to the responding (resp) host. The timestamp *ts* feature indicate the moment of the first packet. The duration of a connection can be calculated by looking at the time between the first and the last packet in a connection.

Table 6.1: Zeek generated identification features

Feature	**Description**	**Type**
uid	Unique identifier of the connection	String
*id.*_h*	IPv4 or IPv6 address of endpoint *	String
*id.*_p*	Port used in endpoint *	String
ts	Timestamp of the first packet	Timestamp
*ts_*	Timestamp of the last packet	Timestamp
duration	Duration of the connection	Deltatime

*={orig,resp}

6.2 Network and Transport Layer

The *conn.log* file consists of data concerning the transport layer protocols TCP and UDP. Zeek also logs the ICMP data, but this protocol is not studied in this research. The log file is divided in two parts: one describing all TCP information and one describing only UDP. This main reason behind this division is the different characterizations of the protocols, as is discussed in in Chapter 2. For UDP, "connections" can be interpreted as sequences of packets going from originator to responder with their corresponding ports. As there are overlapping features between the protocols

concerning bytes and packets transmitted, these features are discussed first in the context of the IP layer. Afterwards, the features that only concern TCP or UDP are discussed.

6.2.1 IP

On IP level, Zeek observes the number of bytes and packets transmitted send over the network. Table 6.2 shows the standard Zeek features concerning these units of information and two new created features. Firstly, the feature $*\_bpp$ is a summary statistic which calculates the average bytes per packet. Secondly, the PCR feature calculates the Producer Consumer Ratio which measures a normalized indication of the information transfer between two hosts. When the ratio is close to one, it is a pure push (e.g. FTP upload), while if the ratio is close to minus one, it is likely a pure pull (e.g. HTTP download) (Bullar & Gerth, 2014).

Table 6.2: IP layer features

Feature	Description	Type
*_*bytes*	Bytes send by endpoint *	Integer
*_*pkts*	Packets send by endpoint *	Integer
*_*bpp*	$\frac{*\_bytes}{*\_packets}$	Float
PCR	$\frac{orig\_bytes - resp\_bytes}{orig\_bytes + resp\_bytes}$	Float

*={orig,resp}

6.2.2 TCP

Recall from Chapter 2 that TCP connections should be established before they can be used. TCP connections are different from UDP as the connection are reliable and connection-oriented. The network layer is, however, not reliable so it occurs that sometimes packets are dropped. When this happens, packets need to be retransmitted. A feature generated by Zeek is the *missing_bytes* feature, which counts the number bytes dropped. This feature might indicate congestion which could indicate malicious behaviour seen on the network. While this feature is useful for the TCP protocol, this feature is meaningless in UDP.

The connection-oriented character of TCP is logged by Zeek using two features: *conn_state* and *history*. The first feature describes the state of the connection at the end of the connection while the latter feature shows the flags observed in the packets transferred in the connection. These two features are depended as flags observed on the wire determine the state of a connection. In the Zeek documentation of the conn.log[1], descriptions are given for each state in the *conn_state*. We believe that recording the events seen in the establishment of a connection separately from the termination might be more appropriate rather than looking at the state of a connection at the end of a connection. Primary reason for this preference is the fact that TCP connections can still be open by keep-alives and later be closed. Misuse could be better detected by observing the behaviour in the establishment and the behaviour of the hosts in the termination of the connection. Therefore, new features are proposed to describe the establishment behaviour between hosts and the termination of the connection. Table 6.3 shows the features which indicate how TCP connections are established. Each of these features describe a situation which could occur in the establishment of a connection. The feature $WEIRD$ combines all other situations which do not belong to the first six cases. Where Table 6.3 describes the opening of a connection, Table 6.4 shows the cases on how TCP connections can be terminated, or remain open. This set of features replaces the Zeek generated *conn_state* and *history* features.

[1] https://docs.zeek.org/en/current/scripts/base/protocols/conn/main.zeek.html

Table 6.3: TCP connection establishment features

Feature	Description
$S0$	No SYN packet observed
$S1$	Only a packet with a SYN flag is observed
$S2$	Originator sends SYN, responder reacts with SYN-ACK, but no final ACK
$S3$	The connection is normally established according to the three-way handshake
$REJ1$	Originator sends SYN packet but replied with a RST packet by the receiver
$REJ2O$	SYN and SYN-ACK are exchanged, but followed by RST send by originator
$REJ2R$	SYN and SYN-ACK are exchanged, but followed by RST send by responder
$WEIRD$	A SYN packet is observed, but none of the above cases were observed

Table 6.4: TCP connection termination features

Feature	Description
$OPEN$	A connection is established, but no FIN or RST flag is observed
$TERM$	Connection gracefully terminated by originator and receiver
$CLSO$	Originator sends a FIN flag but receiver did not respond with FIN or RST
$CLSR$	Receiver sends FIN flag but originator did not respond with FIN or RST
$RSTR$	Receiver abruptly ends connection with a RST flag
$RSTO$	Originator abruptly ends connection with a RST flag

Another feature logged in the conn.log file is the *service* describing which application uses the connection. The port number of the responding host often identify the protocol in usage, e.g. HTTP port 80, FTP port 21, SSH port 22. For the TCP layer, the following services are recorded in the three datasets: HTTP, FTP, SSL, SSH, POP3, SMTP, IMAP, ICR, RFC, DCE/RPC and KRB. For each of these protocols, a binary feature is created indicating whether these protocols are used.

6.2.3 UDP

UDP is an unreliable connectionless protocol. Where *history* and *conn_state* are relevant for TCP connections, in UDP they are meaningless. As hosts are either sending data or they do not, the features *history* and *conn_state* are replaced with features shown in Table 6.5. Just as in the TCP features, the feature *service* is one-hot-encoded for next protocols: DNS, KRB and DHCP. The last feature engineering step in the UDP connection dataset is the transformation of the *orig_ip_bytes* and *resp_ip_bytes*. The minimum size of an IP packet with an empty datagram is 28 bytes for IPv4 (20 bytes IPv4 and 8 bytes UDP) and 48 for IPv6 (40 bytes IPv6 and 8 UDP). As almost in all cases, seen by data analysis, UDP datagrams are empty, two variables are created which indicate whether there is at least one packet in the connection with payload. Therefore, the IP bytes features are replaced with the new features stated in Table 6.5.

Table 6.5: UDP Layer Extracted Features

Feature	Description	Type
$*\_active$	Endpoint * sends at least one packet	Binary
$*\_min\_size$	Endpoint * sends at least one UDP datagram containing payload	Binary

*={orig,resp}

6.3 Application Layer

6.3.1 DNS

In the DNS protocol, clients request address translation from URI to IP addresses or vice versa. As has been shown in Chapter 2, the DNS information exchange consists of a question and an answer. Therefore, to create usable features, we distinguish the creation in three parts: first analysing the DNS header section, then then question section and at last the answer section. In the header section, the AA, RA, RD and TC (Chapter 2) can be used as boolean features indicating whether these flags are used. The *trans_id* only identifies the query to use the answer again when the query reoccurred. As this is random, this is ignored. With *rcode_name* and *rcode* being the same, only *rcode* will be used and is one-hot-encoded using the values from Mockapetris (1987). The question that a host requires an answer to has a *qclass* and *qtype*. These are one-hot-encoded using the recommended values in Mockapetris (1987) and Cheung (2020).

Table 6.6: DNS client and server reply codes

Feature	*	Description	Type
qclass_*	0,1,3,4,254,255	Query class * by client	Integer
qtype_*	1,2,5,6,12,15,16,20,28	Query type * by client	Integer
rcode_*	0,1,2,3,4,5,6	Reply code * by DNS server	Integer

A DNS server responds the question in the answer section. As there could be multiple addresses in one query, the feature *answers_n* counts the number of IP addresses answered by the DNS server. Each answer has a Time To Live (TTL) for which the maximum, mean and minimum values are calculated. When there is no answer, these values will be 0. As hosts can send multiple queries at the same time, the DNS protocol connections are aggregated to prevent double occurrences of the same query in the same connection.

6.3.2 HTTP

The *http.log* file consist of all HTTP messages exchanged between clients and servers. The content of these messages are described in Chapter 2. First, the request line in the request message consists of the method, version and URI. Table 6.7 shows the created features for the method and version. Fielding and Reschke (2014) and Dusseault and Snell (2010) are used as values for the one-hot encoding of $method_*$. The considered HTTP versions are deployed versions in practice.

Table 6.7: HTTP method and version features

Feature	*	Description	Type
method_*	{GET,HEAD,POST, PUT,DELETE,CONNECT, OPTIONS,TRACE,PATCH}	Method * in HTTP request line	Binary
version_*	{0.9,1.0,1.1,2.0}	HTTP version * in usage	Binary

Table 6.8 shows these features constructed for the URI of the resource identifiers, the host and the referrer. Chapter 2 shows that a URI consist of several components. For each component, the length of this component is counted, the number of unique characters and the number of reserved characters in these components are counted. Table 6.9 shows the composed features from the server reply. The reply message of the server consists of the status line and a section of headers. The status line states the *status_msg*, which is a three digit code. To be able to one-hot encode this status message, only the first digit is considered relevant as the other two digits give more explanation conditioned on the first digit.

Table 6.8: HTTP URI features

Feature	Description	Type
*_**_len	Length of the ** part of the *	Integer
*_**_unique_char	Unique characters in the ** part of the *	Integer
*_**_res_char_n	Reserved characters in the ** part of the *	Integer

*={uri,host,referrer}, ** ={netloc,path,params,query,fragment}

Table 6.9: HTTP reply message features

Feature	*	Description	Type
**_mime_types_*	{applicatio, audio, example,font,image, model,text,video}	Endpoint ** sends mime type *	Binary
*xx_code	{1,2,3,4,5,6}	Server responds with status line code *xx	Binary

** ={orig,resp}

6.3.3 FTP

FTP can be used for remote login between client and server to transfer files. Between the client and the server is a data connection and a command connection. Table 6.10 shows the features extracted from the *ftp.log* file. It would be expected that these features are binary, but as the commands are grouped per connection, they are made integer. Each connection has a starting command and a finishing command. The feature *command* shows how often commands are executed over an connection. The *mime* is the type of file that is being transferred over the connection. The reply codes are the responses from the server. These reply codes can be grouped in series (1xx, 2xx, ...) and for each of these series a features can represent the number of times these series are seen on the network.

Table 6.10: FTP created and extracted features

Feature	*	Description	Type
command_∗	{ABOR, CWD, DELE, LIST, MDTM, MKD, NLST, PASS, PASV, PORT, PWD, QUID, RETR, RMD, RNFR, RNTO, SITE, SIZE, STOR, TYPE, USER}	Number of times command * is executed in a connection	Integer
mime_type_∗	{application, audio, example, font, image, model, text, video}	Number of times mime type * is send in one connection	Integer
*reply_code_d*1_∗	{1, 2, 3, 4, 5, 6}	Number of reply code of the first digit is *	Integer
*reply_code_d*2_∗	{0, 1, 2, 3, 4, 5}	Number of reply code of the second digit is *	Integer

6.4 Secure Layer Protocols

The last section of this chapter describes the features representing SSL and SSH usage. Zeek logs for these protocols information concerning cryptographic algorithms, but these features are not taken into account in this study. Primarily reason for this choice is to prevent having many zero-entires by one-hot encoding those categorical features. Nevertheless, there are other features which can be used for machine learning purposes.

6.4.1 SSL

The *ssl.log* file consists of SSL protocol usage. Table 6.11 shows the created and default features extracted from default Zeek features. The *version_∗* and *next_protocol_∗* are one-hot encoded features by using the values of these default features as new features. The second column ∗ indicates which values where used for the one-hot encoding. These are the values observed in the three datasets. The *server* concerning features are extracted by counting some meta-date of the *server_name*. The last three features *established*, *resumed* and *alert* are default Zeek features.

Table 6.11: SSL created and extracted features

Feature	*	Description	Type
version_∗	SSLv2, SSLv3, TLSv10, TLSv11, TLSv12, TLSv13	Indicates which SSL/TLS version is used	Binary
next_protocol_∗	http/1.1, h2	Indicates what the next protocol is	Binary
server_name_dot		Number of dots in the server name	Integer
server_name_dash		Number of dashes in the server name	Integer
server_name_len		Character length of the server name	Integer
server_name_unq		Number of unique characters	Integer
established		Indicates whether the SSH is established	Binary
resumed		Indicates whether the connection is resumed	Binary
alert		Indicates whether there occurred an alert	Binary

6.4.2 SSH

Table 6.12 shows the features created from the *ssh.log* file. As the SSH protocol is mainly used for remote login, relevant features include the number of authentication attempts and whether authentication is successful. The version features indicate whether SSH version 1 or 2 is used. Other Zeek generated features are combined by counting the number of missing values per connection. Therefore, the lacking values are summarized in the missing values counter.

Table 6.12: SSH created and extracted features

Feature	Description	Type
auth_attempts	Number of authentication attempts	Integer
auth_success	Authentication result (1 Accepted, 0 Failed)	Binary
ssh_version_1	Indicates whether SSH version 1 is used	Binary
ssh_version_2	Indicates whether SSH version 2 is used	Binary
missing_values	Counts the number of missing values	Integer

Chapter 7

Feature Analysis

Where Chapter 6 describes the features constructed for this research, this Chapter provides an analysis of the whole dataset. The focus of this chapter is directed on meta-data, class distributions and the relationships between features. First, some facts including number of instances and features about the prepared dataset are given. Second, the class distributions of the datasets are stated to put the occurrences of malicious traffic in proportion with normal traffic. At last, some correlations are calculated to give insight in the dependency between features.

7.1 Meta-data on Selected Datasets

After creating some new features described in Chapter 6, there is one protocol specific dataset for each day of each experiment. To work toward having only one protocol dataset for each of the three selected studies, the daily internet traffic datasets are aggregated over the days. This step resulted eventually in 21 different datasets; each describing one protocol for each study. Table 7.1 shows the number of connections, or so to say instances, and the total number of features. The number of features here describe the number of explanatory variables used to predict the class feature. It can be observed that each dataset has different number of instances for the considered protocols. In the UNSW-NB15 dataset, the number of SSL connections is for example redundant while the HTTP usage in ISCX-IDS-2012 has exploded. As HTTP utilizes the TCP protocol as underlying transport layer protocol, it is expected that when HTTP usage increases the number of TCP connections also grow. The periods in which these datasets are captured can be read in Chapter 5.

Table 7.1: Number of instances and features for each dataset

Protocol	CIC-IDS-2017	ISCX-IDS-2012	UNSW-NB15	Features
TCP	1,074,723	2,021,661	1,491,982	41
UDP	1,043,725	642,827	566,072	17
DNS	1,071,911	878,273	405,072	45
HTTP	534,042	8,837,111	377,705	90
SSL	332,982	47,467	182	19
SSH	8,254	11,933	50,542	7
FTP	2,476	1,673	46,630	50

7.2 Class Analysis

Where Table 7.1 shows the number of connections for each protocol, Table 7.2 shows the intrusion ratio for each protocol. The intrusion ratio is defined as the number of malicious instances divided by the total number of instances. The bold intrusion ratio's indicate that these intrusion ratios are unrealistic high. In practice, it would be unreal if, for example, 50% of all traffic is malicious. Therefore, the bold intrusion ratio's are randomly down sampled to an intrusion ratio of 5% for the anomaly detection algorithms, which seems more realistic. To prevent excluding attacks, the downsampling is performed with the restriction that there has to be at least one instance for each attack. As the first question in the introduction is to check whether malicious activities are indeed different, the datasets before the downsampling are used for the supervised learning techniques. In the next sections, a class analysis is shown for the three datasets.

Table 7.2: Intrusion ratio per dataset

Protocol	CIC-IDS-2017	ISCX-IDS-2012	UNSW-NB15
TCP	**50%**	3.74%	3.61%
UDP	0.05%	0.0012%	2.31%
DNS	0.01%	0.0009%	4.34%
HTTP	**51.65%**	**44.12%**	**6.74%**
SSL	0.0003%	0%	**92.86%**
SSH	**37.02%**	**42.14%**	0.04%
FTP	0.32%	0.24%	2.68%

7.2.1 CIC-IDS-2017

Before taking a look at the class distributions, it is interesting to see the attack scheme of the datasets. Figure 7.5 shows the attack scheme of the CIC-IDS-2017 dataset. During five sequential days, five hours of network traffic were captured and during these days as set of attacks were executed. The first day of the experiment consists of only normal traffic, but other days consist of both normal and malicious traffic. The port scan is the only attack performed during two days.

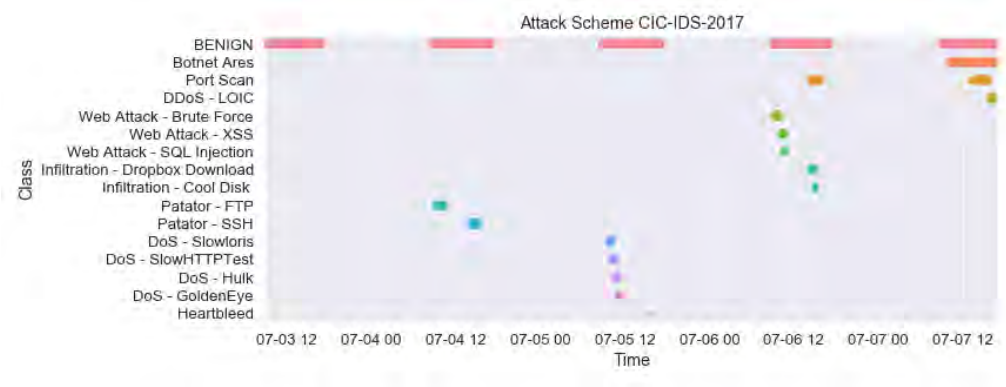


Figure 7.1: CIC-IDS-2017 TCP layer attack scheme

The previous figure has shown the periods in which the attacks were performed seen on the TCP dataset of the CIC-IDS-2017 experiment. Figure 7.2 shows the class distribution of the normal and the set of malicious classes of this dataset. The intrusion ratio is approximately 50%. The CIC-IDS-2017 dataset published by Sharafaldin et al. (2018) consists of both TCP and UDP connections. In this research, TCP and UDP connections are separated, so the class distributions are different. Obviously, port scans, DoS and DDoS attacks create many connections by their inherent

characteristics. DoS and DDoS attacks aim at flooding hosts with large numbers of connections and port scans need to check many ports to check if a port is open. In contrast, smaller attacks, such as the heartbleed, requires less connections.

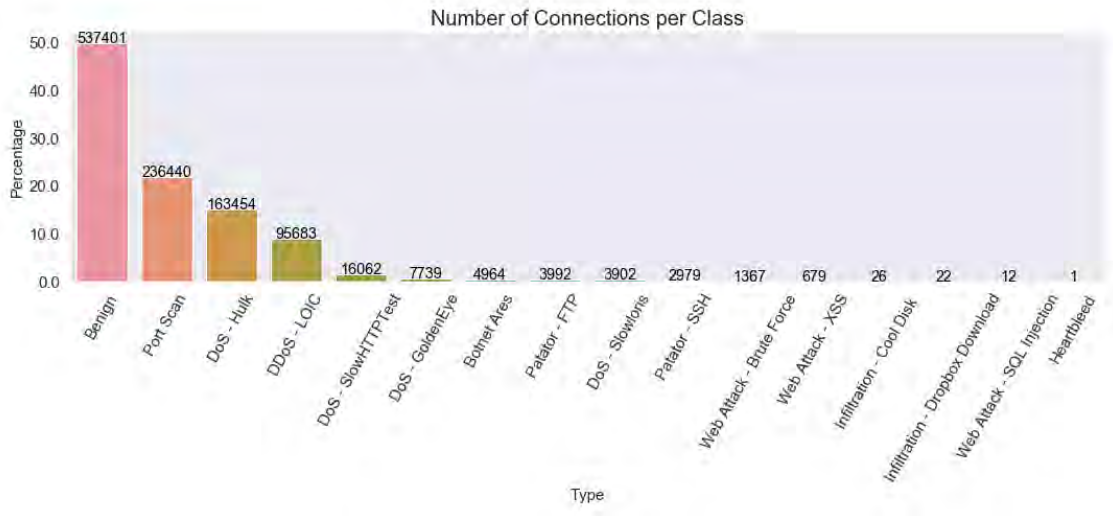


Figure 7.2: CIC-IDS-2017 TCP layer class distribution

Figure 7.3 shows the class distribution of the HTTP layer. In the attack scheme of the CIC-IDS-2017 it can be observed that most HTTP attacks are executed on the third day of the experiment. The HTTP class distribution is similar to the TCP class distribution as most attacks occur using HTTP which of course uses TCP. A major difference between HTTP and TCP classes lies in the distribution of the port scan attack. In the TCP layer is the port scan a dominant class, but this does not hold for the HTTP dataset. The port scan attack focuses on multiple ports and not necessarily HTTP, which usually uses port 80.

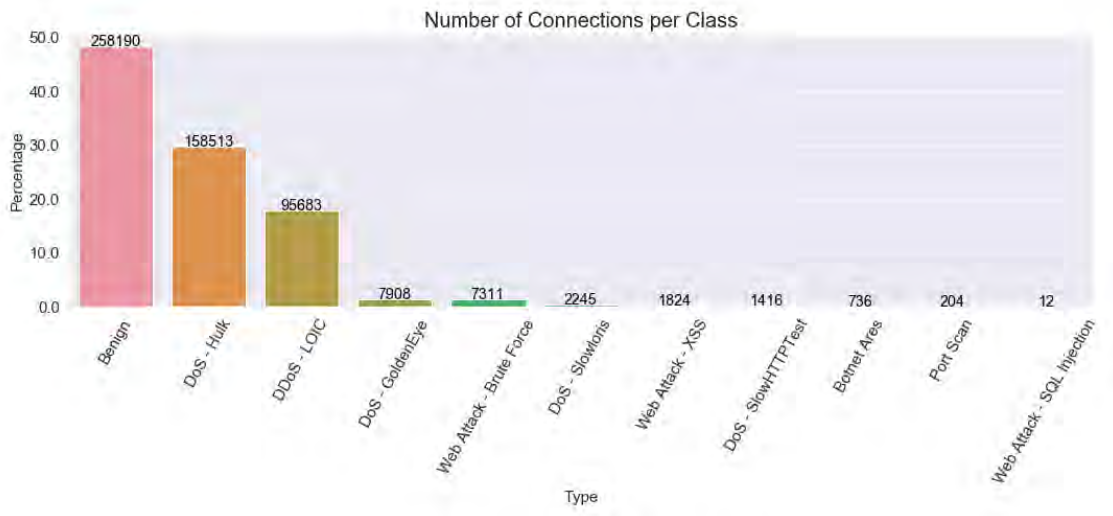


Figure 7.3: CIC-IDS-2017 HTTP layer class distribution

Port scan attacks can be observed on multiple protocol layers. This attack is the only attack observed on the DNS, FTP and UDP with only 3165, 8 and 3165 instances respectively. When

looking at the experiment description of the CIC-IDS-2017, it would be expected that there are also occurrences of the Patator - FTP attack in the FTP protocol. That is however not the case and it is unclear why this is the case. On the SSL protocol there is only one connection with the Heartbleed attack. Other instances are benign for this protocol. The last protocol is the SSH protocol. Figure 7.4 shows the class distribution of the SSH protocol, where both the Patator - SSH and the port scan attack are logged.

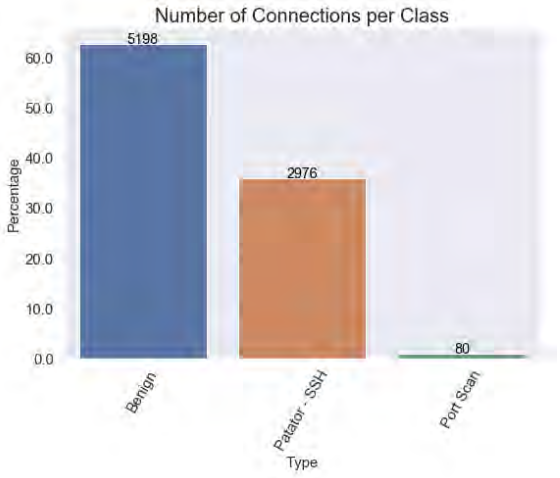


Figure 7.4: CIC-IDS-2017 SSH layer class distribution

7.2.2 ISCX-IDS-2012

The ISCX-IDS-2012 has a different attack scheme with different attacks compared to the CIC-IDS-2017 dataset. Figure 7.5 gives a schematic view of the attack moments. Most attacks occur in a small interval of time, while one attack (the so-called backdoor) remains open until the end of the experiment. With this backdoor, the attackers have the ability to send commands to the program which was installed during the infiltration attack. Using these commands, the attacker could perform several attacks on different hosts.

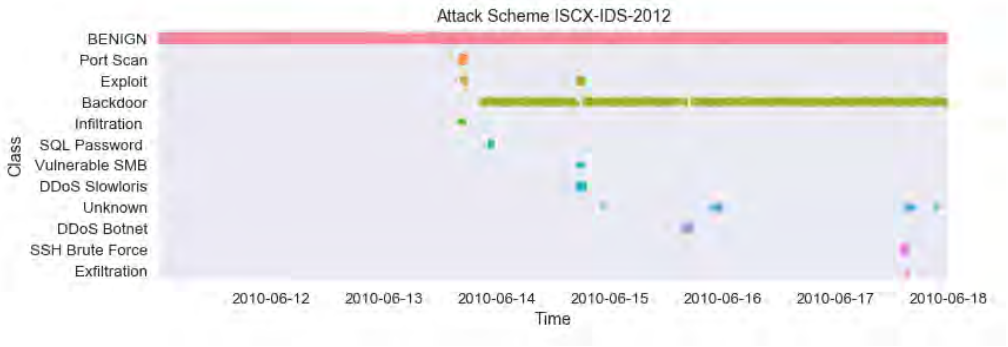


Figure 7.5: ISCX-IDS-2012 TCP layer attack scheme

Figure 7.6 shows the class distribution of benign traffic and each attack separately for the TCP layer. It can be observed that almost all traffic is normal and a small amount is malicious. This class distribution seems to be a better representation of reality in which only a small part of the network traffic is actually malicious. When looking at the attacks, the attacks with the highest

number of connections are the DDoS attacks and port scans.

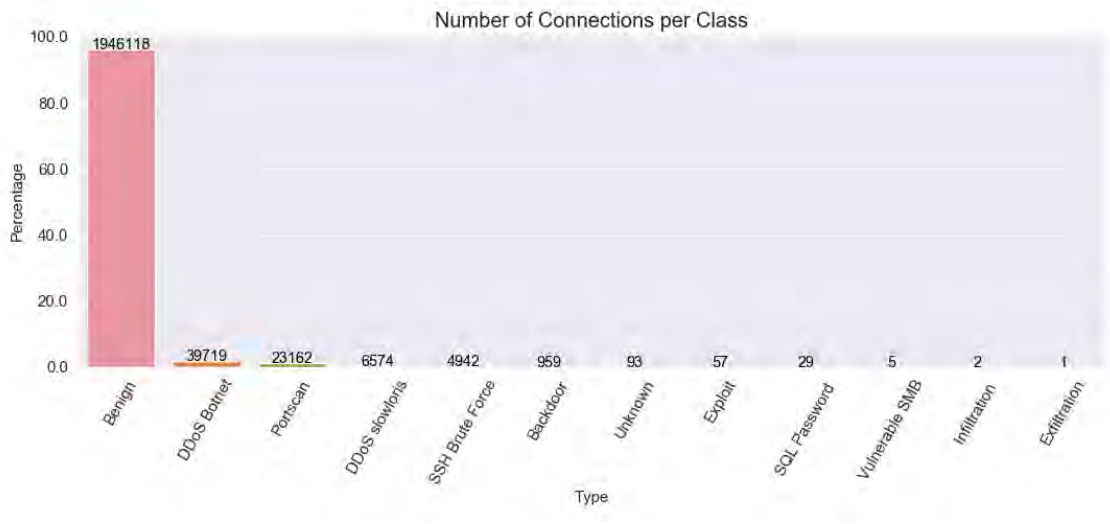
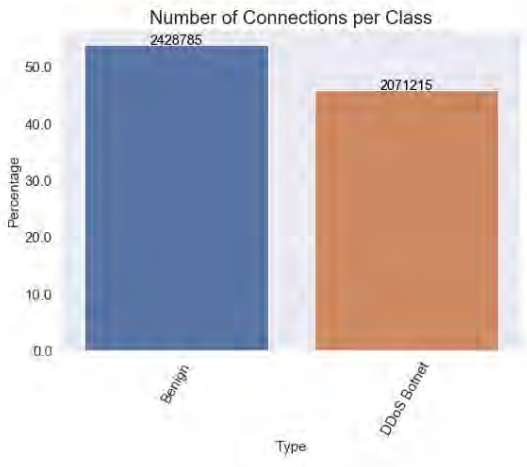
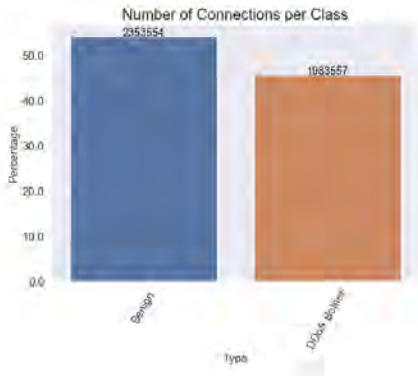


Figure 7.6: ISCX-IDS-2012 TCP class distribution

In the ISCX-IDS-2012 dataset, the HTTP dataset protocol contains more than eight million connections. With this large number of connections and the large number of features, the HTTP file is split in two parts to make computation possible. The first dataset contains the chronological first 4.5 million connections, while the second part contains around 4.3 million. Figure 7.7 shows the type distribution of the two HTTP datasets. The intrusion ratio is unrealistically high as more than 40% of the instances are malicious.



(a) Part 1



(b) Part 2

Figure 7.7: ISCX-IDS-2012 HTTP layer class distribution

Similar to the CIC-IDS-2017 dataset, the ISCX-IDS-2012 dataset UDP, DNS and FTP only contain port scan attacks. For these protocols, there are only 8, 8 and 4 port scan instances respectively. The SSL protocol is ignored as there are no malicious activities in this dataset. Figure 7.8 shows the distribution of the attacks on SSH protocol. As has been said, there are some unknown attacks which cannot be identified based on the description of the attacks in the paper of the ISCX-IDS-2012 dataset. The exfiltration attack is the attack where data is extracted from the victim host

after performing the SSH Brute Force attack.

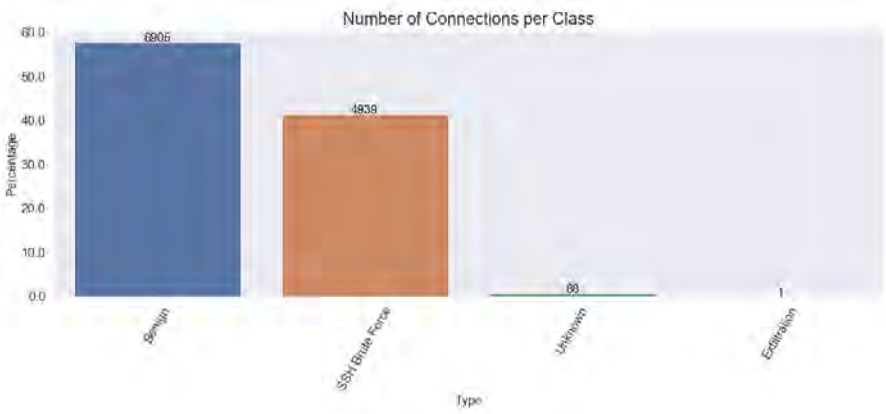


Figure 7.8: ISCX-IDS-2012 SSH class distribution

7.2.3 UNSW-NB15

In contrast to the datasets, we observe that in attack scheme of the UNSW-NB15 dataset, shown in Figure 7.9, are not limited to a certain moment of the day. All attacks are performed the entire experiment. This is of course an unrealistic scenario as networks are not always constantly engaged by attackers. Still, this dataset is usefull as attacks can be observed on multiple protocols rather than one single protocol, which is discussed next.

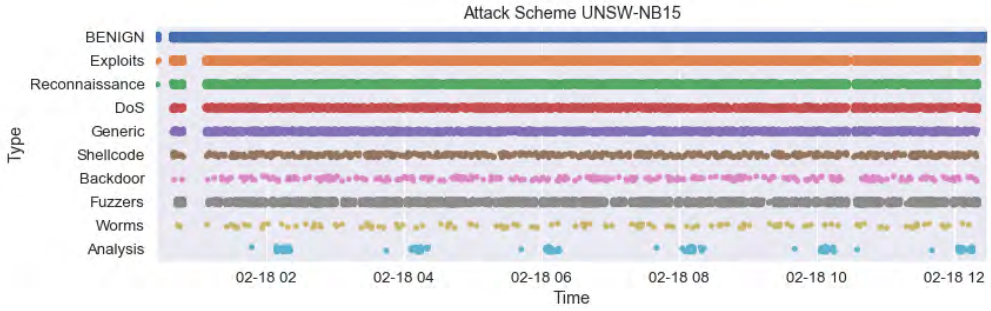


Figure 7.9: UNSW-NB15 TCP layer attack scheme 18 Februari

Table 7.3 shows the class distributions of all classes for each protocol. Similar to the ISCX-IDS-2017 dataset, the intrusion ratio seems realistic. However, most attack connections are not a DDoS or Portscans, but exploits, fuzzers or fuzzers. In contrast to previous datasets, attacks are more distributed over multiple layers rather than a few. Portscan attacks was the only class in previous datasets which transcend several attacks. The SSL protocol does not resemble realistic internet traffic, as almost all instances are malicious. There are only 13 normal instances, so this protocol is not considered relevant for further analysis.

Table 7.3: UNSW-NB15 class distribution all protocols

Class	TCP	UDP	DNS	HTTP	FTP	SSH	SSL
Analysis	434	0	0	428	0	0	0
Backdoor	315	34	0	229	0	0	0
Benign	1,438,143	552,995	387,512	352,249	45,380	50,522	13
DoS	3,153	412	151	2,004	26	0	21
Exploits	24,757	607	385	17,149	1,036	14	66
Fuzzers	14,393	5,430	3,362	1,337	183	0	0
Generic	3,061	1,000	13,615	2,087	5	6	82
Shellcode	750	761	0	0	0	0	0
Reconnaissance	6,827	4,82	47	2,087	0	0	0
Worms	149	21	0	149	0	0	0

7.3 Feature Analysis

This section highlights feature statistics for all datasets. It shows where the datasets differ in normal protocol usages and some correlations between the features. For this analysis, only the normal traffic is considered to describe the differences.

7.3.1 TCP

The first protocol for feature analysis are the TCP datasets. Figure 7.10 shows the TCP connection and termination features for the three datasets. It can be observed that in the UNSW-NB15 almost all connections are correctly opened and closed. The correctly establishment and termination of TCP connections ratio is less for the other two datasets. In the ISCX-IDS-2012 dataset, most connections are only closed by the originator but not replied by the responder. Furthermore, hosts do often not reply to establishment attempts from originators in this dataset.

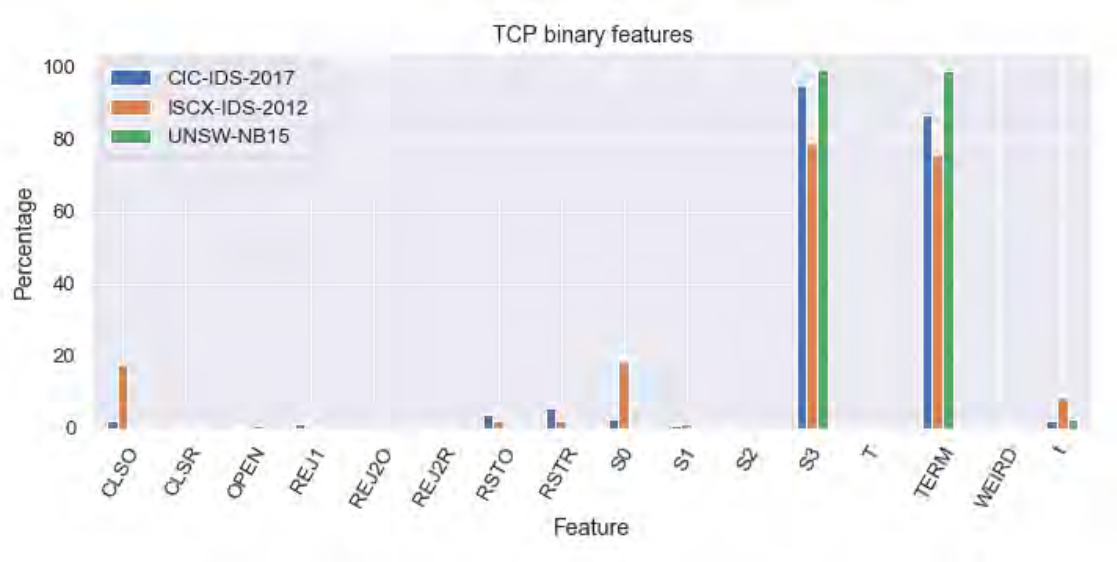


Figure 7.10: TCP connection establishment and termination features

Each experiment has own protocol usage schemes. Figure 7.11 shows which services are used using the TCP protocols.UNSW-NB15 uses a broad variety of protocols, even more than Zeek has regristed. Most normal behaviour in the ISCX-IDS-2012 is HTTP traffic. The CIC-IDS-2017 is

primarily used for SSL protocols usage rather than HTTP which indicates that HTTPS is used over HTTP.

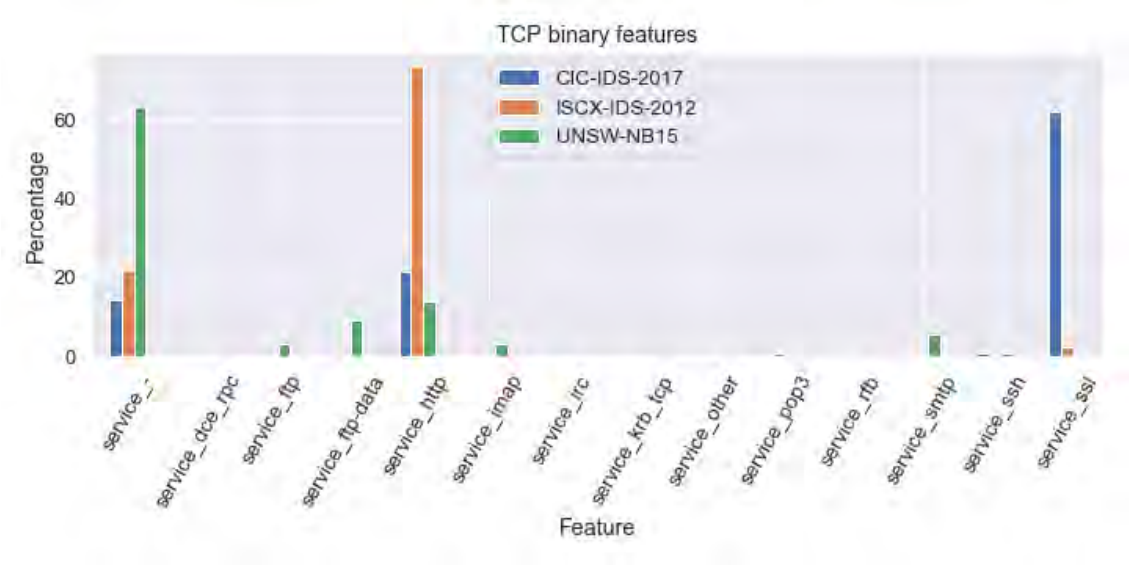


Figure 7.11: TCP service features

Figure 7.12 shows the boxplots of the PCR and the durations of the TCP connection. Excluded from these figures are the points outside of the whiskers. A reason for this choice is that there are many points that are outliers so showing these points would make the box plot unclear. It can be observed that the ISCX-IDS-2012 dataset has a lower PCR ratio, which means that there are more pull requests than push requests. This is inline with the observation that most TCP traffic uses HTTP in the ISCX-IDS-2012. When looking at the duration of connections, TCP connections are much longer for the CIC-IDS-2017 dataset. This is inline with the fact that in this dataset most TCP traffic uses SSL as application layer protocol.

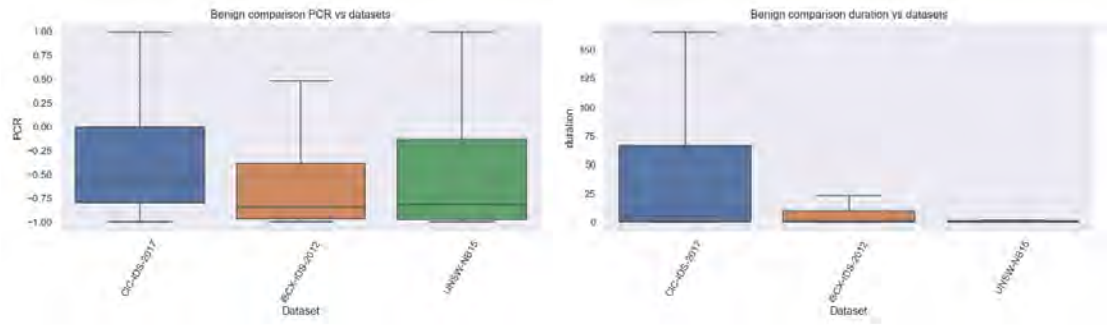


Figure 7.12: TCP PCR and duration features

7.3.2 UDP

The following protocol which is analysed is UDP. Figure 7.20 show the Spearman correlation between the features of UDP. It can be seen that the bytes and packets features are heavily monotonic correlated. This makes sense as more bytes to send require more packets. When looking at the service features, it is expected that these are not correlated as they are one-hot encoded.

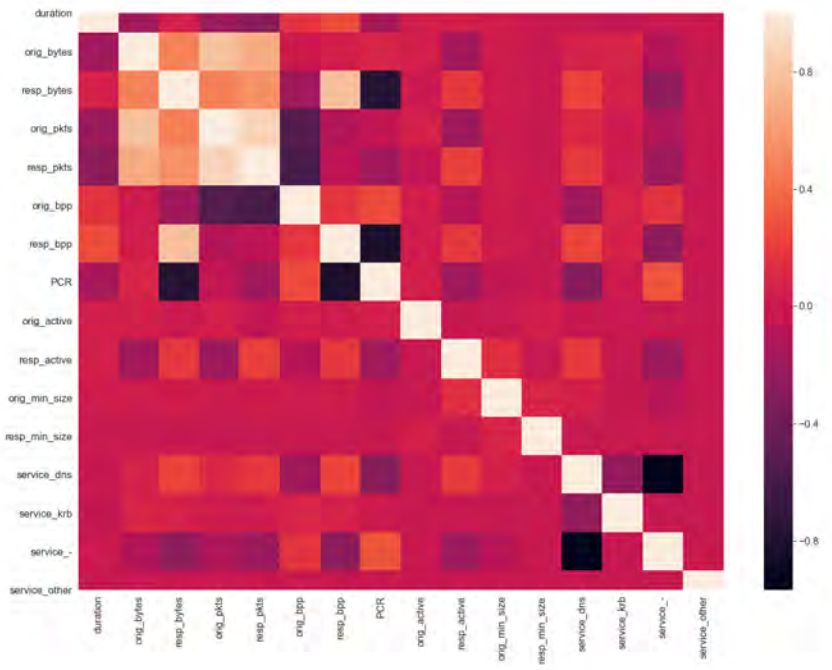


Figure 7.13: CIC-IDS-2017 UDP Spearman correlations

The differences between UDP and TCP can be shown by looking at the difference in duration. Figure 7.14 shows the PCR and duration (in miliseconds) of UDP connections. Comparing the durations of UDP and TCP connection show that UDP connection periods are almost redundant. The PCR is very similar between UDP and TCP.

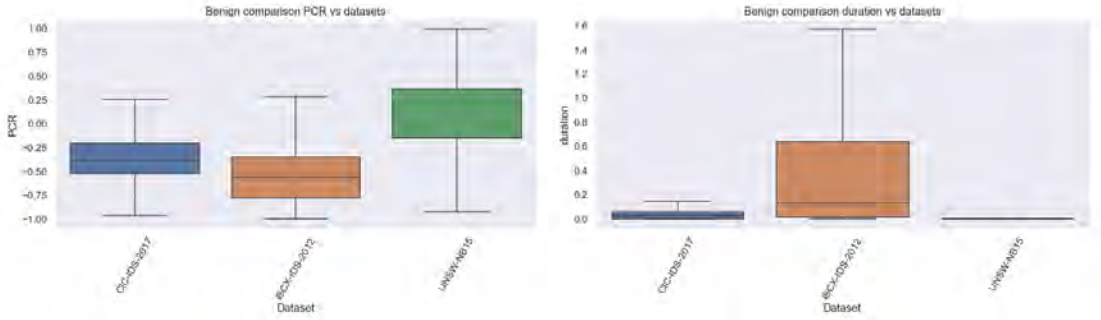


Figure 7.14: UDP PCR and duration features

7.3.3 HTTP

If clients want web pages from web servers, the client can send for example HTTP request to web servers. Clients and servers have to agree upon the used HTTP protocol before the request can be replied. This negotiation can be performed in the request line of the request message. Figure 7.15 shows which HTTP versions were used in the three experiments. As HTTP 1.1 has keep-alives, it is expected that the experiments dominantly use the HTTP 1.1 version.

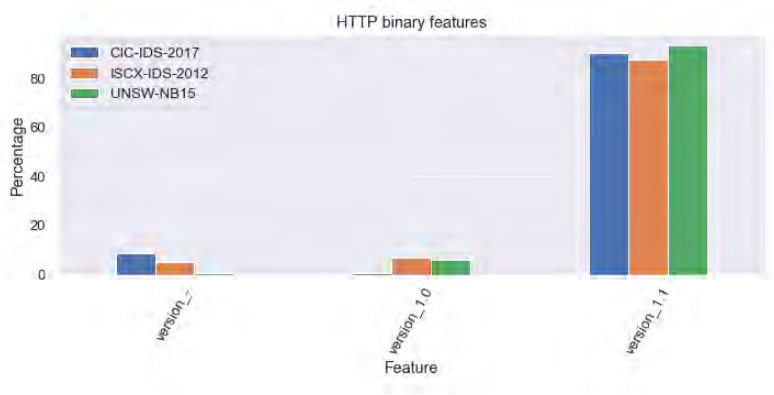


Figure 7.15: HTTP server reply codes

After a HTTP request is send, the server will reply with a status code to the client. Figure 7.16 shows which reply codes occur most in the normal traffic behaviour. In most cases, the server replies that the request is OK and the web page can be send to the client. In the UNSW-NB15 however, there are many client errors. As can be observed, by the simulation of internet traffic, many client errors can be made.

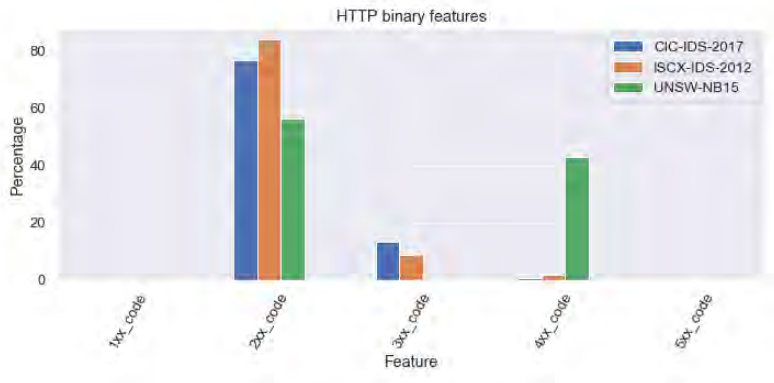


Figure 7.16: HTTP version usage

7.3.4 DNS

The DNS protocol can be analyzed by looking at the DNS header section, the question section and the answers provided by the DNS (name)server. Figure 7.17 shows the flags used in the heaer section. The RA and RD flags are almost always turned on for all DNS queries. In addition, the TCP usages and the rejected DNS queries are given, which are almost 0 for normal traffic.

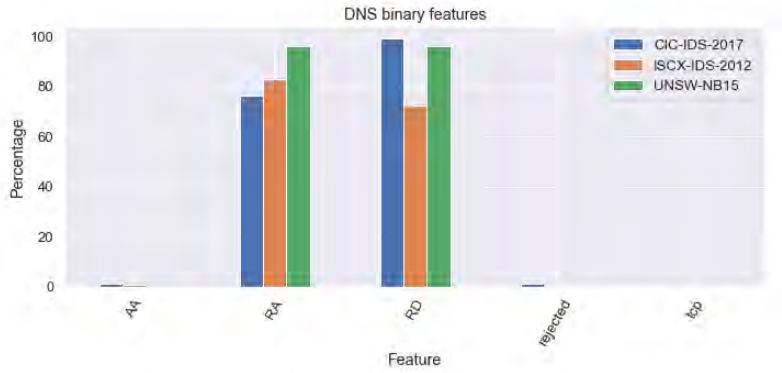


Figure 7.17: DNS flag

The query section consists of the query type and the query class. Figure 7.19 shows the query classes usages and Figure 7.19 gives the query type usages. The query classes are dominantly of class 1. Other classes do not occur that often. A similar analysis can be applied on the qtype of the DNS requests. Only qtype 28 is often searched, which is the IPv6 address lookup.

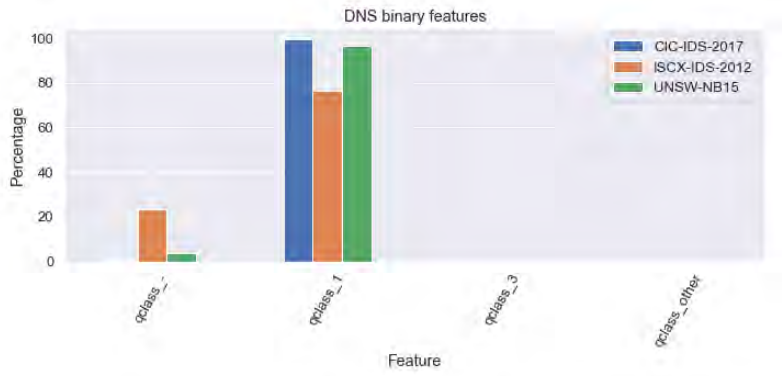


Figure 7.18: DNS qclass

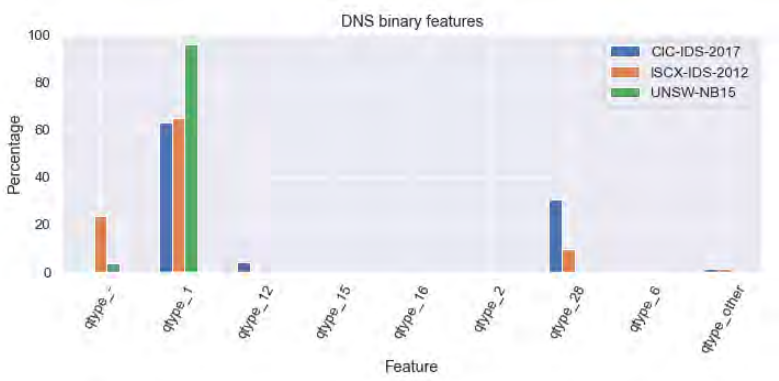


Figure 7.19: DNS qtype

7.3.5 FTP

When looking at the Spearman rank correlations between the FTP features given in Figure 7.20, there are some positive monotonic relationships observable and some negative monotonic relationships. It is expected that there are some expected reactions from the server when certain commands are executed by the client. This implies that there are some positive or negative relationships between the constructed features. These existing relationships do not necessarily mean that the dataset is unusable, but it is of course something to take into account when interpreting the results on the datasets considering this protocol.

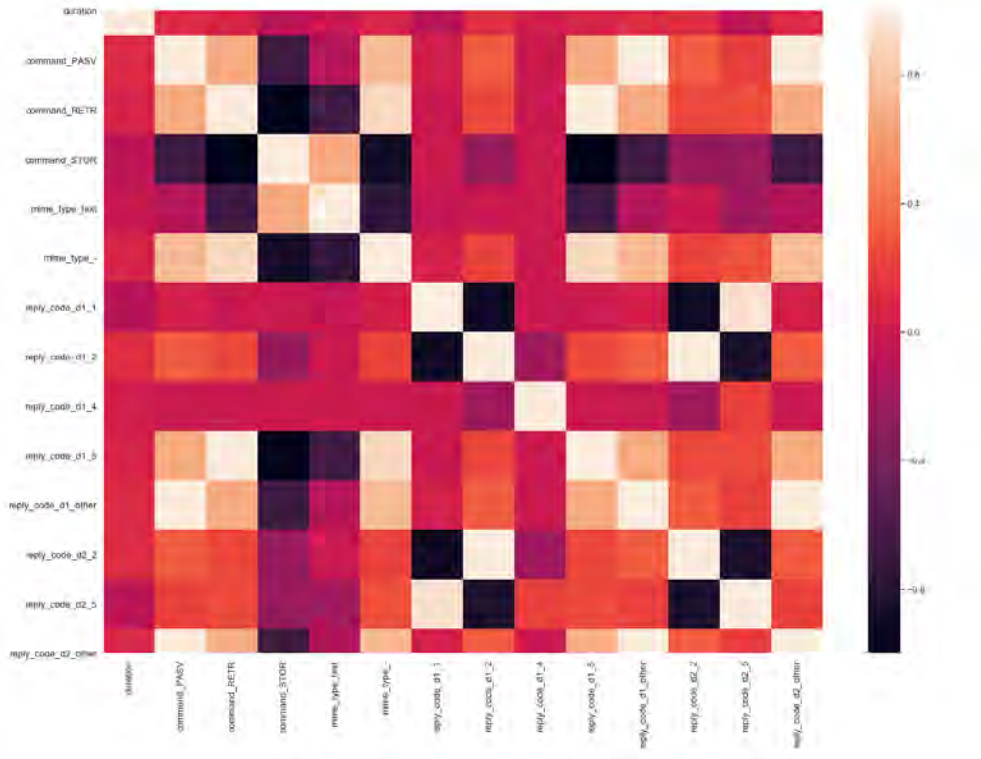


Figure 7.20: CIC-IDS-2017 FTP Spearman correlations

7.3.6 SSL

The next protocol is the SSL protocol. Figure 7.21 shows whether hosts where able to establish ssl connections, whether alerts occurred and whether SSL connections where resumed afterwards. For the ISCX-IDS-2012 dataset it is more clear that SSL connections where more used compared to the other datasets. The number of alerts is redundant for normal traffic.

Figure 7.21: SSL specific features

In the establishment of a SSL connection, the hosts need to agree on the version in use. Figure 7.22 shows the versions used over the different experiments. The ISCX-IDS-2012 is only using TLS version 1.0, while the UNSW-NB15 uses that version and the SSLv3 as SSL version. In the CIC-IDS-2017 the improved TLS version 1.2 is consistenly used to create a secure line. Important to notice that only 13 connections actually use the SSL protocol in the UNSW-NB15 experiment.

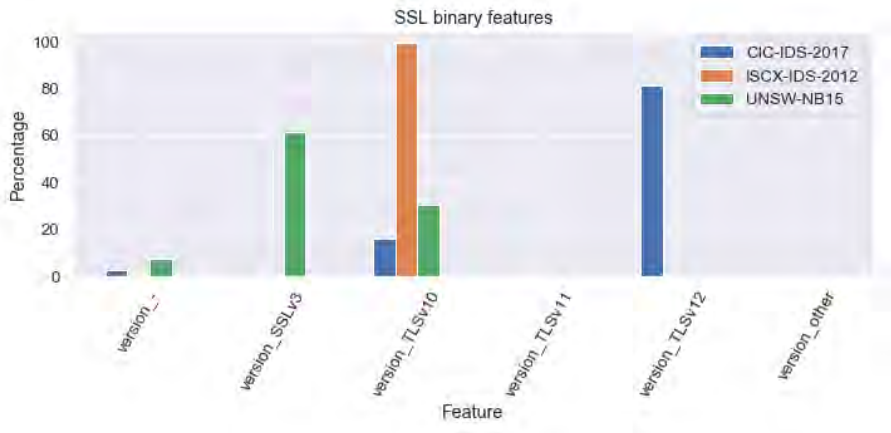


Figure 7.22: SSL version features

7.3.7 SSH

The last protocol which is discussed in this chapter is the SSH protocol. Figure 7.23 shows the binary features of the SSH datasets. It can be observed that for all three experiments there is a preference for using the SSH version 2 above the version 1. While for the CIC-IDS-2017 the authentication is almost always successful, for the UNSW-NB15 this is never the case. It is more likely that almost all authentication trials are successful when only looking at normal traffic.

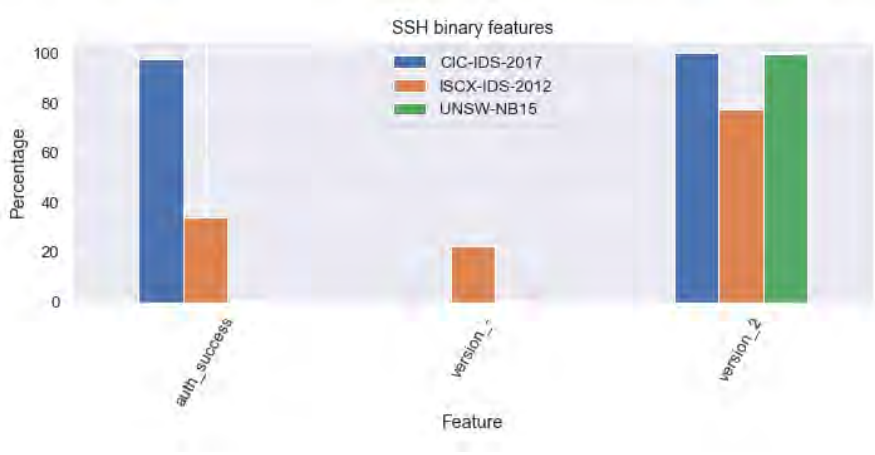


Figure 7.23: SSH binary features

The previous figure has shown that almost none of the UNSW-NB15 normal traffic authentication trials were successful. Figure 7.24 shows the number of remote login trials on the SSH protocols. It is unlikely that each SSH remote login is successful in the first attempt. However, in the CIC-IDS-2017 the number of attempts is always 1 or 0. This unlikely in practice. In contrast, the UNSW-NB15 dataset has authentication attempts with more than 100 trials. Neither this is likely when considering normal internet traffic.

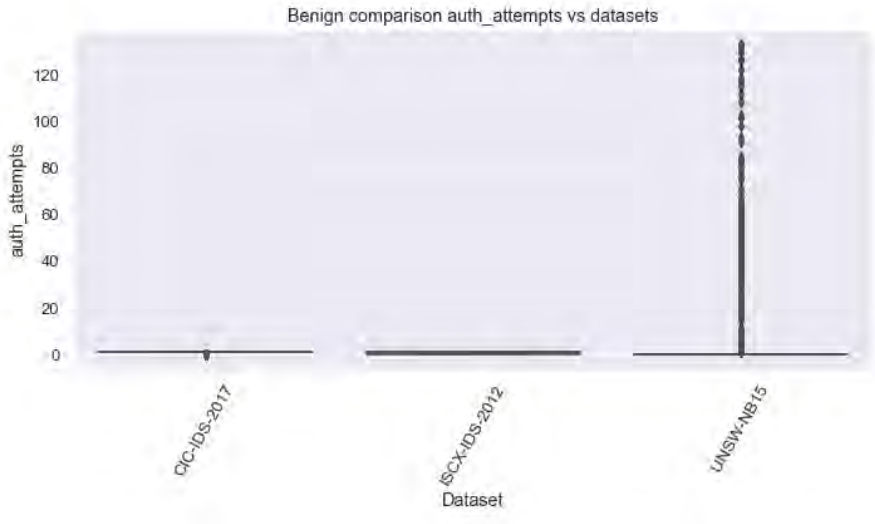


Figure 7.24: SSH authentication attempts

Chapter 8

Methodology

This chapter discusses the methods used to conduct this research. In the first part of this chapter, an overview of the mathematical notation is given. Besides that, the important distributions that are used in this chapter will be stated. In the second part of this chapter, we will start with the supervised learning methods to detect intrusions by using the actual labels. With these methods, it can be checked if indeed machine learning techniques are able to distinguish intrusions from normal traffic. Afterwards, the anomaly detection techniques are described. These methods assume that the underlying labels of the connections are unknown. At last, the graph-based methods are stated which will be used to detect anomalous timestamps in dynamic changing networks.

8.1 Mathematical Notation

Before diving in the methods, some mathematical notation should be clear for the reader to understand what we mean with certain variables. As different fields have different mathematical notation, it is good to have variable definitions. When we use capitalized variables, such as X we mean matrices while with small letters we mean vectors or constants.

- X : matrix with response variables in the columns
- x_i : i-th observation where $i \in \{1, 2, ...n\}$
- x_j : j-th feature where $j \in \{1, 2, ..., m\}$
- x_{ij} : the ith entry of the matrix of the jth feature
- y : response variable with label $k \in C = \{C_1, C_2, ..., C_K\}$
- $\hat{y}$: prediction of an arbitrary instance
- $\hat{y}_i$: prediction of instance i
- s : anomaly scores vector
- s_i : anomaly score instance i

8.2 Distributions

In this chapter, two statistical distributions are used. The first distribution is the binomial distribution. This is a sum of Bernoulli variables in the sense that suppose $X_i \sim Bern(p)$ for $i = 1, ..., n$, then it follows that $X = X_1 + X_2 + ... + X_n$ is binomial distributed with parameters n and p. The probability function of this variable is given by:

$$\mathbb{P}(X = k) = \binom{n}{k} p^k (1-p)^{n-k}$$

The expectation of this binomial distribution is given by $E(X) = np$ and its variance is $\sigma^2(X) = np(1-p)$. The second distribution which is used in this section is the hypergeometric distribution. In this distribution, we randomly draw d samples from a population of size n containing P positives and $n - P$ negatives. The hypergeometric distribution describes the probability on drawing k positive samples from this distribution. Its probability is given by $X \sim h(d, P, n)$:

$$\mathbb{P}(X = k) = \frac{\binom{P}{k}\binom{n-P}{d-k}}{\binom{n}{d}}$$

8.3 Supervised Learning Algorithms

The assumption that malicious traffic is indeed different from normal traffic needs to be verified. Several considered supervised learning algorithms are described in this chapter to check this assumption. These algorithms can be used to learn the characteristics of the attacks.

8.3.1 Gaussian Naive Bayes

The Naive Bayes classifier is build upon Bayes theorem for conditional probabilities (Aggarwal, 2015a). The model is called "naive" in the sense that it assumes that the features are independent. The naive Bayes model belongs to the probabilistic classifiers. Say X is an observation with an unknown class label. Let H_i be the hypothesis which indicates that instance X belongs to class C_i. Then the posterior probability $P(H_i \mid X)$ indicates the probability that this hypothesis holds given this observed observation X. Bayes theorem states that this posterior probability can be calculated using the following formula.

$$\text{Posterior} = \mathbb{P}(H_i \mid X) = \frac{\mathbb{P}(X \mid H_i) \cdot \mathbb{P}(H_i)}{\mathbb{P}(X)} = \frac{\text{Likelihood} \cdot \text{Prior}}{\text{Evidence}}$$

In a classification problem, we are interested in finding this posterior probability $\mathbb{P}(y_i = C_k \mid x_i)$ that determines the probability that observation i belongs to class C_k given data instance x_i. Using Bayes theorem, this would result in finding the following probabilities.

$$\mathbb{P}(y_i = C_k \mid x_i) = \frac{\mathbb{P}(x_i | y_i = C_k) \cdot \mathbb{P}(y_i = C_k)}{\mathbb{P}(x_i)} \propto \mathbb{P}(x_i \mid y_i = C_k) \cdot \mathbb{P}(y_i = C_k)$$

As can be observed, the evidence is excluded from the equation as this evidence is the same for each class. Therefore, finding this probability is reduced to finding the likelihood and the prior probability. Finding the prior probability that data instance i belongs to class C_k can be derived a priori. In the naive Bayes model, the likelihood is calculated by assuming all features are conditional independent. By this assumption, the likelihood function can be derived by taking the product of all individual probabilities.

$$\mathbb{P}(x_i \mid y_i = C_k) = \prod_{j=1}^{m} \mathbb{P}(x_{ij} | y_i = C_k)$$

To determine these individual probabilities, an underlying distribution of the feature values is required. These are the so called event models of the naive Bayes classifier. There are commonly three distributions often used as event models: Bernoulli, multinomial and Gaussian. In the Bernoulli event model, the feature is required to be Bernoulli distributed. The multinomial model requires feature to represent frequencies. In the Gaussian event model, the features are assumed to be normally distributed. For the Gaussian model, the individual probabilities can be calculated using the following formula.

$$\mathbb{P}(x_{ij}|y_i = C_k) = \frac{1}{\sqrt{2\pi\sigma_{j,k}^2}} exp\left(-\frac{(x_{ij} - \mu_{j,k})^2}{2 \cdot \sigma_{j,k}^2}\right)$$

Here, the parameters $\sigma_{j,k}$ and $\mu_{j,k}$ are estimated using the maximum likelihood estimator for each feature j and each class k.

$$\mu_{j,k} = \frac{\sum_{i=1}^{n} x_{i,j} \cdot \mathbb{1}_{y_i=C_k}}{\sum_{i=1}^{n} \mathbb{1}_{y_i=C_k}}, \quad \sigma_{j,k}^2 = \frac{\sum_{i=1}^{n} (x_{i,j} - \mu_k)^2 \cdot \mathbb{1}_{y_i=C_k}}{\sum_{i=1}^{n} \mathbb{1}_{y_i=C_k}}$$

After computing the posterior probability of each class for an data instance i, the remaining question is how to predict this instance. The naive Bayes model makes use of the maximum a posteriori (MAP) decision rule to determine which class will be predicted. The MAP decision rule indicates that the class with the highest posteriori probability is used to predict data instance i. The following objective is used to predict this instance.

$$\hat{y}_i = \arg\max_{C_k} \mathbb{P}(y_i = C_k) \prod_{j=1}^{m} \mathbb{P}(x_{i,j} \mid y_i = C_k)$$

While this model is very intuitive, the conditional independence assumptions the model is almost never valid. In most situations there are dependencies between features which are ignored by the model. Nevertheless, the model is often used in NLP settings. Furthermore, in the Gaussian event model the underlying assumption is that the features are Gaussian distributed. As this does not have to be the case, the model can under perform. Furthermore, for the Bernoulli and the multinomial the model does not work well with categorical data that is not observed in training the model.

8.3.2 Nearest Neighbour

A second algorithm is the nearest neighbour algorithm (Murty & Devi, 2011). In contrast to other classifiers, training the model is not performed in the training phase, but in the testing phase. These classifiers are so called lazy learners. The rationale behind the lazy learner is that similar instances have similar class labels. In the K nearest neighbours classifiers, the labels of the K nearest instances of an testing instance are used to predict the label of that testing instance. Let us first define the distance between two instances i and l as $d(x_i, x_o)$. Then for example the Minkowski distance of order q between these instances is defined as:

$$d(x_i, x_o) = (\sum_{j=1}^{m} \mid x_{i,j} - x_{o,j} \mid^q)^{\frac{1}{q}}$$

Here, when q is 2, then we are working with the Euclidean distance. This algorithm is instances based in the sense that the instances are compared with each other and the nearest K neighbours from one instance are used to predict the label of an instance. Define A_i as the set of training instances which are closest to test instance i. Then when each instance has an equal weight the prediction becomes the following. The easiest method is brute force. This requires however $O(n_{test} \cdot n_{train} \cdot m)$

After finding the A_i, we can make a prediction for each sample.

$$\hat{y}_i = \arg\max_{C_k} \sum_{i=1}^{n} \mathbb{1}_{y_i = C_k \wedge i \in A_i}$$

Another option is to weight each sample with the distance between the points:

$$\hat{y}_i = \arg\max_{C_k} \sum_{i=1}^{n} \frac{1}{d(x_i, x_{\hat{i}})} \cdot \mathbb{1}_{y_i = C_k \wedge i \in A_i}$$

This algorithm does not require fitting as there are not parameters to tune. The algorithm is expensive in time in the testing part as instances are compared with the trained data in order to make a prediction. When the nearest neighbours are found, based on the majority vote the labels are predicted.

8.3.3 Decision Tree Classifier

A decision tree is a tree classifier in which the classification process is executed by dividing observations in a tree structured manner. In this tree, each node represents a test on a attribute and each branch represent the outcome of that test.

The decision tree algorithm works by choosing an feature and splitting the observations on the basis of some decision rule. This procedure is performed in a top-down approach in such way that each internal node can be split in two branches. There are two criteria to split the observations: on their Gini index or the Entropy. The Gini index is a criterion to minimize the probability of misclassification, while entropy is a way to measure impurity. The following equations show how these criterion can be calculated. Here, P_j is the probability on class j.

$$Gini = 1 - \sum_j p_j^2, \quad Entropy = -\sum_j p_j log_2 p_j$$

We split on the feature that results in the largest information gain (IG). the information gain is defined as:

$$IG(D_p) = I(D_p) - \frac{N_{left}}{N_p} I(D_{left}) - \frac{N_{right}}{N_p} I(D_{right})$$

where I is either the entropy, Gini index or classification error.
Table 8.1 shows other parameters for the the decision tree classifier. Each parameter determines limitations on the tree size.

Table 8.1: Parameters decision tree

Parameter	**Description**
max_depth	The maximum depth of the tree
min_samples_split	The minimum number of samples required to split an internal node
min_samples_leaf	The minimum number of samples required to be at a leaf node
max_features	The number of features to consider when looking at the best split

8.3.4 Adaboost Classifier

Adaboost is a boosting technique to combine weak learners into a strong classifier. These weak learners are classifiers that perform poorly on a classification task. Each of the weak learners are trained on a random subset of the total training set. AdaBoost assigns a weight to each training example which determines the probability that each example should appear in the training set. After .... increases the weight of misclassifying example so that these become a larger part of the training set such that the next classifier performs better.

Boosting is the process of combining weak learners into a strong learners (Freund & Schapire, 1997). With weak learners we mean a classifiers which is inaccurate in predicting the correct classes. Adaboost initializes a weight vector which defines the probability being chosen for learning a weak learner.

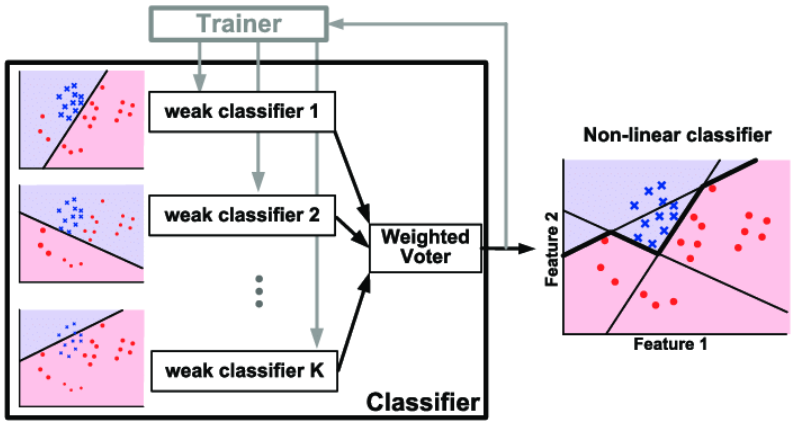


Figure 8.1: Illustration of the Adaboost classifier Wang et al. (2015)

8.3.5 Random Forest Classifier

A random forest classifier combines multiple classification trees by sampling instances from one dataset (Breiman, 2001). By sampling instances with or without replacement from the original dataset, sub-samples are constructed for which new trees are build. The random forest classifier therefore is an ensembled classifier in the sense that it combines multiple classifiers in one. There are several procedures to construct the sub-samples for one tree. It can be performed with or without replacement. When chosen with replacement, this means that samples are bootstrapped. Suppose that trees are trained on a sub-sample which is constructed without replacement and we sample the same number of instances as in the original dataset, the original dataset is constructed. This means that the

Suppose that we want to train one tree on a sub-sample. Then the decision tree classifier is trained on the dataset. There are some choices that can be made in the construction of the tree. For example, not all features are required to be taken into account for making a split. This can be limited. To make a prediction, each sample is predicted by each tree. Each tree has the same weight so the end prediction is done by the majority vote scheme. Figure 8.2 gives an visual illustration how prediction are made. Each tree votes for an certain class and based on majority vote the final prediction is made. As each tree is different, each tree have different nodes and thus have different branches.

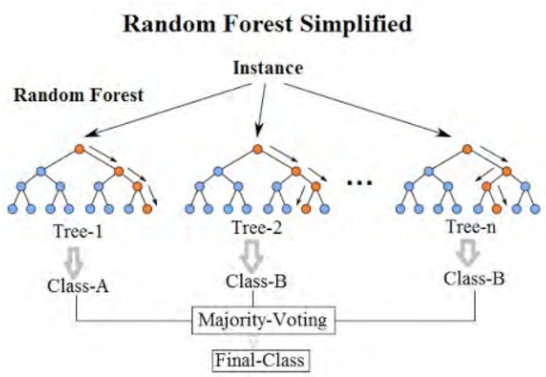


Figure 8.2: Random Forest Example (Will Koehrsen, 2017)

8.4 Anomaly Detection Techniques

In this section, the anomaly detection algorithm are discussed which are used to determine the anomaly scores for the connections. First, the principle component analysis is described and afterwards the isolation forest. These algorithms do not use the actual labels of the connection to determine the anomaly scores.

8.4.1 Principal Component Analysis

Using principle component analysis(PCA) as an anomaly detection technique is proposed in Shyu, Chen, Sarinnapakorn, and Chang (2003). Observing that intrusion detection datasets are often high dimensional in nature and the goal of PCA is to reduce dimensionality makes PCA a desirable method. PCA is concerned with explaining the variance-covariance structure by linearly transforming datasets into a new dataset. This transformed dataset consisting of principle components has three important properties, a) the components are uncorrelated, the first component has the highest variance, b) the second component has the second highest variance, and so on, and c) the total variation of all principle components is equal to the total variation in the original dataset. In the approach of Shyu et al. (2003), anomaly scores can be calculated by constructing a hyperplane which captures most of the variance in the data and calculating the difference between instances and this hyperplane.

To construct the hyperplane, the covariance matrix should be calculated which indicates the variances and covariances between the response variables. Suppose we have the a mean-centered dataset X. This mean-centred dataset is the original dataset where each feature is mean-centred. Then the variance-covariance matrix can be calculated as follows.

$$\Sigma = \frac{X^T X}{n}$$

On the diagonal of this matrix are the variances of the features while on the other places are the covariances. A property of this Σ matrix is that it is square $(m \cdot m)$, symmetric and positive semi-definite. With these properties, the matrix can be diagonalized in the following way (Aggarwal, 2015b).

$$\Sigma = PDP^T$$

The matrix P consists of the orthonormal eigenvectors of Σ and D is a diagonal matrix containing the non-negative eigenvalues. The same trick can be applied on the correlation matrix R where all features are standardized.

$$D = \begin{bmatrix} \lambda_1 & 0 & 0 & ... & 0 \\ 0 & \lambda_2 & 0 & ... & 0 \\ 0 & 0 & \lambda_3 & ... & 0 \\ ... & ... & ... & ... & ... \\ 0 & 0 & 0 & ... & \lambda_m \end{bmatrix}, P = [e_1, e_2, ..., e_m]$$

The eigenvectors and eigenvalues, $(\lambda_1, e_1), (\lambda_2, e_2), ..., (\lambda_m, e_m)$, are sorted in such way that $\lambda_1 \geq \lambda_2 \geq ...\lambda_m$. By this, it is certain that the first component explains most variance. Now it holds that $\sum_{j=1}^{m} \lambda_j = p$ when the correlation matrix is used instead of the covariance matrix. To get the transformed dataset, the original dataset X can be multiplied with the matrix P.

$$X' = XP$$

Each row of this new transformed matrix consists of the principle component values for the instance of that row. Define $v_{i,j}$ as the j-tj principle component value of the i-th instance. Then the anomaly score for an instance can be calculated by:

$$s_i = \sum_{j=1}^{m} \frac{v_{i,j}^2}{\lambda_j}$$

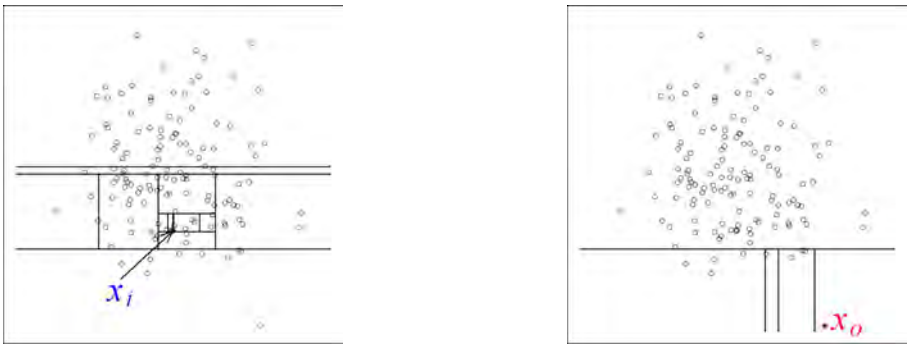
It is possible that all principle components can be used. But, it is also possible to only look at the anomaly scores of the principle components with the highest variance or the lowest variance. The developed method in (Shyu et al., 2003) looks at the anomaly scores determined by looking at a model with the highest explained variances in combination with a model with the lowest explained principle components. In the model with the highest variances, the top 50% of the explained variance are used while for the lowest principle components an eigenvalue of at most 0.2 is used. In the machine learning setting, the eigenvalues and eigenvectors can be learned using the training data. These values can later be applied on the test data to extract the transformed dataset with less principle components if required.

8.4.2 Isolation Forest

A second method which is considered as anomaly detection technique is Isolation Forest (iForest) (Liu, Ting, & Zhou, 2008). In contrast to other model-base approaches in anomaly detection, iForest explicitly isolates anomalies instead of profiling normal behaviour. This method is an ensemble method as it utilizes a set of Isolation Trees (iTree) where each tree tries to isolate instances. Isolation in this context means that instances are separated from all other instances. Each iTree isolates instances by randomly dividing the data space until all instances are isolated.

Partitioning the instances is performed using a proper binary tree where each node is either an external node with no child or an internal node with two daughter nodes. In each internal node, a test is performed on a random attribute using a random split value to divide the instances in the two daughter nodes. This random attribute is chosen from a set of attributes for which the instances values are different. In an external node, there is either i) only one instance, which means the instance is isolated, ii) all instances have the same values so there is no attribute to split the instances on or iii) the maximum tree height is reached (Liu et al., 2008).

As anomalies are considered to be "few and different", it is expected that the path between the anomaly instances in the nodes of the iTree and the root of the iTree are shorter than the paths to normal instances. Figure 8.3 illustrates how iForest isolates two instances x_i and x_o in a two dimensional space. Each line represents the split value on which the data space is divided. As can be observed, x_i requires more splits to be isolated compared to x_o. This means that the path of the normal instance x_i is longer than the path to anomaly instance x_o.



(a) Isolating x_i (b) Isolating x_0

Figure 8.3: Isolation Forest in Action (Liu et al., 2008)

Now that is shown how instances are isolated in an iTree, let us look at how the anomaly scores are determined. The anomaly scores depend on the path length of the instances in the iTree. Suppose $h(x)$ is the path length of a point x in the tree. Then this path length is the minimum number of edges x traverses from the root node to the node x. Then the anomaly score of instance x is given by:

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}}$$

Here, n is the number of instances, $E(h(x))$ is the expected path length, which is the average $h(x)$ of an collection of iTrees. By using more iTrees, the expected $h(x)$ is more certain. $c(n)$ is the average path length of unsuccessful search in a binary search tree.

$$c(n) = 2 \cdot H(n-1) - 2 \cdot \frac{n-1}{n}$$

The $H(i)$ is the harmonic number and can be approximated by $\ln(i) + 0.5772156649$. Figure 8.4 shows the relationship between the expected path length and the anomaly score. When the expected path length is close to 0, it is definitely an anomaly and get a score of 1. When the path length is close to the number of instances, then is almost certainly normal behaviour.

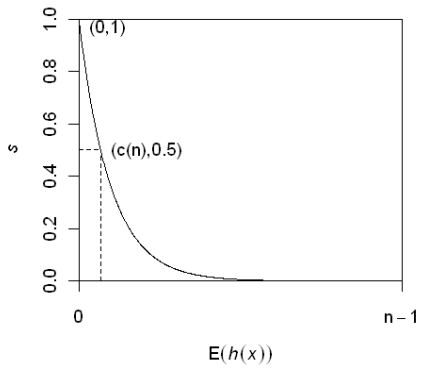


Figure 8.4: Expected path length and anomaly score relationship (Liu et al., 2008)

8.5 Graph Based Anomaly Detection Techniques

In this last section, we describe the techniques for anomaly detection in graph-structured datasets. When merely looking on connection level, the communication dependencies between hosts is forgotten. Rather than looking at connections, we can also look at the connections from a network perspective. There are many metrics to compare two graphs. Graph distance metrics can be distinguished in two types of metrics: weighted and unweighted. In unweighted metrics, link usage is binary, while in weighted metrics each link has an certain weight. Firstly, some graph definitions are given for mathematical formality. Afterwards, four considered unweighted graph metrics and four weighed graph metrics are given.

8.5.1 Definitions

Let us define a set of Edges E and a set of nodes N for graph G. Furthermore, let us define G_t as the graph on timestamp t where $t \in \{0, 1, ..., T\}$. This means that we have a series of graphs which evolve over time. By looking at the change over time, we can compute the change of the network over time. In a graph G, E is the set of edges in that graph and V is the set of nodes. $w_E^G(u, v)$ is the weight of the edge between nodes u and v.

Maximum Common Sub-graph

The maximum common sub-graph (MCS) is the fully connected graph with the most number of edges between two graphs. This graph can be constructed by combining two graphs into one single graph and determine the strongly connected sub-graphs with the most edges.

8.5.2 Unweighted Graph Distance Metrics

Vertex/Edge Overlap (VEO)
Two graph are similar is they share similar edges and nodes. In the Vertex/Edge Overlap distance, the Jaccard index is computed to measure these similarities between two graphs. The intersection of edges and nodes is weighted by the total number of edges and nodes of both graphs (Papadimitriou et al., 2010).

$$d_{VEO}(G, H) = 1 - 2 \cdot \frac{\mid V_G \cap V_H \mid + \mid E_G \cap E_H \mid}{\mid V_G \mid + \mid V_H \mid + \mid E_G \mid + \mid E_H \mid}$$

Network Editing Distance (NED)
The Network or Graph Editing Distance between two graphs is calculated by evaluating the number of edit operations required to transform graph G to graph H (Showbridge, Kraetzl, & Ray, 1999).

$$d_{NED}(G, H) = \mid V_G \mid + \mid V_H \mid - 2 \cdot \mid V_G \cup V_H \mid + \mid E_G \mid + \mid E_H \mid - 2 \cdot \mid E_G \cup E_H \mid$$

MCS Edge Distance (MCSED)
The MCS edge distance is calculating by first finding the similarity between two graphs. This similarity is calculated by counting the number of edges in the MCS and dividing this number by the maximum number of edges in both graphs. This similarity is then subtracted to get a distance metric (Pincombe, 2005).

$$d_{MCSED}(G, H) = 1 - \frac{\mid mcs(E_G, E_H) \mid}{max(\mid E_G \mid, \mid E_H \mid)}$$

MCS Vertex Distance (MCSVD)
This measure is similar to the MCSED but instead of looking at the edges the vertices are compared (Pincombe, 2005).

$$d_{MCSVD}(G,H) = 1 - \frac{\mid mcs(V_G, V_H) \mid}{max(\mid V_G \mid, \mid V_H \mid)}$$

8.5.3 Weighted Graph Distance Metrics

In the following metrics, the weights of the graphs are taken into account. The idea is to use the anomaly scores of the connections which as assigned by the anomaly detection algorithms as weighted for the weighted graph distance metrics.

Weighted Distance (WD)
In the weighted distance metric, the sums of the absolute weight difference is taken divided by the maximum of the maximum of those weights. This sum is then divided by the total number of edges in the intersection of the compared graphs (Pincombe, 2005).

$$d_{WD}(G,H) = \frac{\sum_{u,v \in V} \frac{|w_E^G(u,v) - w_E^H(u,v)|}{max(w_E^G(u,v), w_E^H(u,v))}}{\mid E_G \cup E_H \mid}$$

MCS Weighted Distance (MCSWD)
In the MCS weighted distance, the weighted distance is taken, but only the edges of the MCS are considered (Pincombe, 2005).

Umeyama Distance (UD)
The Umeyama distance is defined as the sum of squares of the difference in the edge weights (Dickinson & Kraetzl, 2003).

$$d_{UD}(G,H) = \sum_{u,v \in V} \left(w_E^G(u,v) - w_E^H(u,v)\right)^2$$

Entropy Distance (ED)
Instead of taking the default weights, weights can be scaled by taking all weights in the network in consideration (Gupta, Gao, Aggarwal, & Han, 2014). In the Entropy distance, the weights are scaled by the sum of the weights over the graph in the following way.

$$\hat{w}_E^G(u,v) = \frac{w_E^G(u,v)}{\sum_{u,v \in V} w_E^G(u,v)}$$

The Entropy of a graph can than be calculated using the following formula.

$$E(G) = \sum_{u,v \in V} \hat{w}_E^G(u,v) \cdot \ln(\hat{w}_E^G(u,v))$$

Given the calculated Entropies for two graphs. The distance between the graphs can be calculated by taking the difference between the difference of the graphs.

$$d_{ED}(G,H) = \mid E(G) - E(H) \mid$$

8.6 Evaluation Methodologies

There are several validation methodologies to evaluate classifiers. Aggarwal (2015a) state that a dataset should be divided in three part: a model building part, a model selection part by parameter tuning and a testing part. This division is illustrated in Figure 8.5. There should be a clear distinction between model selection and model evaluation to avoid having too positive biased evaluations. As such, there are two considered dataset division techniques in this research. These are the holdout method and the cross-validation validation methods.

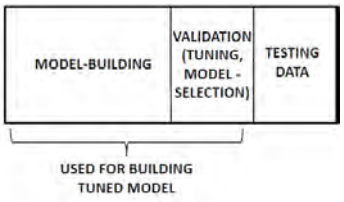


Figure 8.5: Train test split for parameter tuning and evaluation (Aggarwal, 2015a)

8.6.1 Holdout Method

In the holdout validation method, the dataset is randomly split in two sets. One of this dataset is used as training dataset while the other is the test dataset. The largest dataset is used as training dataset while the smaller one is used as test dataset. To get an confidence interval of the performance of the model, this procedure can be performed multiple times when non-deterministic models are used. A stratified approach can be used to preserve that training data class distribution and test class distribution are (almost) equivalent (Charte, Rivera Rivas, Del Jesus, & Herrera, 2016).

8.6.2 Cross-Validation

Another validation technique is cross-validation. In this technique, the data is divided in parts and each part is used to test the model. In this research the interest is in the K-fold cross validation, illustrated in Figure 8.6. Each fold is used once as test dataset and K-1 used as training data for the model to train on.

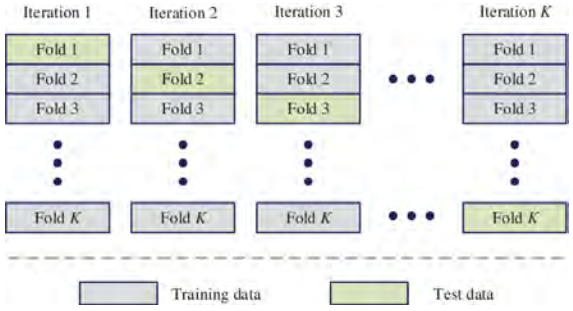


Figure 8.6: K-fold cross-validation (Ren et al., 2019)

For the supervised learning techniques, each fold gets a prediction and when all observations have a prediction, the outcomes of these predictions can be turned into the confusion matrix, which we will discuss in the next section. By computing this Cross-validation techniques multiple times, an estimation can be computed.

In the anomaly detectors, K-Fold can also be applied. Each instances gets an anomaly score as it is tested by in the test set. In the end, all instances can get sorted by their anomaly scores and it can be observed how good the model performs. By doing multiple runs with different folds, a good estimation can be made on the anomaly score of an instance given by an anomaly detector. So suppose each instance gets an anomaly score in a fold. We can take the average anomaly score as score to evaluate the model.

$$\hat{s}_i = \frac{\sum_{i=0}^{L} s_i}{L}$$

8.7 Evaluation Metrics

In practice, classifiers are assigned to make predictions on whether connections are malicious or not. We require metrics to measure whether the predictions are correct. In machine learning, *precision* and *recall* are often used metrics to measure the quality and quantity of the classification. Another metric that is often used for anomaly detection is the F_1 score, which is the harmonic mean of the precision and recall. As datasets in intrusion detection are imbalanced with intrusions underrepresented, the F_1 metric is primarily focussed on labelling these intrusions correctly rather than focussing on normal connections. In this section, the calculation of these metrics in the binary classification and the multi class classification problem are shown. It is shown how these metrics are calculated. Furthermore, to put these metrics in perspective, we discuss two approaches by Jan Klein and Joris Pries to construct baseline to compare the actual classifiers with. In the approach of Jan Klein, each instances is predicted with a Bernoulli distribution in the binary classification problem and a categorical distribution in the multi class classification problem. On the other hand, Joris Pries randomly assigns a predefined set of labels to a dataset.

8.7.1 Binary Classification

In a binary classification problem, each instance has either a ground truth positive (+) or negative (−) label. The goal of this problem is to accurately predict these instances with their corresponding ground truth label. In this binary case, each instance is evaluated by a classifier and gets either a positive (P) or a negative (N) predicted label. These predictions are summarized in a confusion matrix. This is a special contingency matrix as there are only two classes: positives and negatives.

Suppose there are n instances for which the ground truth labels are known. From these instances, P instances are positive and N are negative such that $n = P + N$. Now, the predictions of a classifier can be stored in the confusion matrix shown in Table 8.2. In the rows of the matrix are the actual labels while the columns show the predictions. Each value in this matrix is the count of the number of instances with the actual class given in the row name and the prediction given by the column name.

Table 8.2: Confusion matrix binary classification

Actual\Predicted	**Negative (N)**	**Positive (P)**	**Total**
Negative (−)	True Negative (TN)	False Positive (FP)	N
Positive (+)	False Negative (FN)	True Positive (TP)	P

In this confusion matrix, the False Positives value is the number of Type 1 errors, while the False Negatives value is the number of Type 2 errors. There is a natural trade-off between these types of errors. Making less Type 1 errors implies making more Type 2 errors. Related to these kind of errors are the Recall and Precision evaluation metrics. These evaluation metrics are often used to evaluate classifiers in the field of Information Retrieval.

Binary Metrics

The *precision* is an evaluation metric which shows the fraction of labelled positives which turn out the be actually positive.

$$\text{Precision} = \frac{TP}{TP + FP}$$

In contrast, the *recall* is defined as the percentage of ground truth positives which are labelled positive.

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{TP}{P}$$

As these measure are opposite, there exists a trade-off between them. This trade-off is however not necessarily monotonic. Therefore, taking the average of these metrics is not an appropriate choice. A better metric to summarize these metrics is the F_1 measure, which takes the harmonic mean of these measures.

$$F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} = \frac{2 \cdot TP}{TP + FP + P}$$

One critical note on the F_1 measure is that is ignores the performance of predicting the negative classes correctly (Powers, 2011). In the field of cybersecurity, there is more interest in detecting intrusions rather than predicting whether traffic is non-malicious, so this critical note does not apply in this setting.

Within the binary F_1 score, there is a bias in predicting positives correctly as the number of true negatives (TN), which is the number of correctly labeled negative observations, is no factor in this metric. The F_1 is bounded between 0 and 1, where 1 is the highest score and 0 the lowest. However, there is not a fixed threshold of the score for which the model can be perceived as good or bad. Therefore, we require simple models that can serve as a threshold, or baseline, so that we can compare the outcome of other models with the performance of this model. There are two simple models which are used as baseline: coin flip and random select.

Baseline Binomial

Jan Klein proposes a simple baseline to which we will compare our results with for the binary classification problem is a coin flip classifier. For each observation, a random coin will be tossed with probability θ on throwing heads (class 1) and probability $1 - \theta$ on tails (class 0). This means that we have $TP \sim B(P, \theta)$ and $FP \sim B(N, \theta)$.

$$\mathbb{E}(\text{Recall}) = \mathbb{E}\left(\frac{TP}{P}\right) = \frac{\mathbb{E}(TP)}{P} = \frac{\theta \cdot P}{P} = \theta$$

The expected recall depends on the θ parameter. The expected precision of a random flip coin can be expressed as follows.

$$\mathbb{E}(\text{Precision}) = \mathbb{E}(\frac{TP}{TP + FP}) = \mathbb{E}(\frac{TP}{Z}) = \sum_{z=1}^{n} \sum_{x=0}^{min\{z,P\}} \frac{x}{z} \mathbb{P}_{X,Z}(X = x, Z = z)$$

With $TP + FP = Z \sim Bin(n, \theta)$. If we now use Bayes theorem.

$$= \sum_{z=1}^{n} \sum_{x=0}^{min\{z,P\}} \frac{x}{z} \mathbb{P}_{X,Z}(X = x | Z = z) P_Z(Z = z) = \sum_{z=1}^{n} \frac{1}{z} \mathbb{P}_Z(Z = z) \sum_{x=0}^{min\{z,P\}} x \cdot \mathbb{P}_{X,Z}(X = x | Z = z)$$

The conditional probability $\mathbb{P}_{X,Z}(X = x|Z = z)$ is the hypergeometric distribution with population size n, z draws from with P are positive and finding the probability on x successes. This implies that the latter summation is the expected value of the hypergeometric distribution.

$$= \sum_{z=1}^{n} \frac{1}{z}\mathbb{P}_Z(Z = z) \cdot z \cdot \frac{P}{n} = \frac{P}{n}\sum_{z=1}^{n} P_Z(Z = z) = \frac{P}{n}(1 - \mathbb{P}_Z(Z = 0)) = \frac{P}{n}(1 - (1 - \theta)^n)$$

Using this formula, the highest expected precision can be obtained when θ is 1. The expected F_1 score can also be obtained for this classifer. Again using $TP+FP = Z \sim Bin(n, \theta)$ and the Bayes Theorem, the same trick is applied.

$$\mathbb{E}(F_1) = \mathbb{E}(\frac{2 \cdot TP}{2 \cdot TP + FP + FN}) = 2 \cdot \mathbb{E}(\frac{TP}{Z + P}) = 2 \cdot \sum_{z=0}^{n} \sum_{x=0}^{min\{z,P\}} \frac{x}{z + P} \cdot \mathbb{P}_{X,Z}(X = x, Z = z)$$

$$= 2 \cdot \sum_{z=0}^{n} \sum_{x=0}^{min\{z,P\}} \frac{x}{z + P}\mathbb{P}_{X,Z}(X = x|Z = z)P(Z = z) = 2 \cdot \sum_{z=0}^{n} \frac{\mathbb{P}(Z = z)}{z + P} \sum_{x=0}^{min\{z,P\}} x \cdot \mathbb{P}_{X,Z}(X = x|Z = z)$$

$$= 2 \cdot \sum_{z=0}^{n} \frac{1}{z + P} \cdot \mathbb{P}(Z = z) \cdot z \cdot \frac{P}{n} = 2 \cdot \frac{P}{n} \cdot \sum_{z=0}^{n} \frac{z}{z + P} \cdot \mathbb{P}(Z = z)$$

Now, we are interested in finding the optimal score. Let us take a look at some inequalities. The following inequality holds as $\frac{x}{x+a}$ is a increasing function in x for $x \geq 0$.

$$\mathbb{E}(F_1) = 2 \cdot \frac{P}{n} \cdot \sum_{z=0}^{n} \frac{z}{z + P} \cdot \mathbb{P}(Z = z) \leq 2 \cdot \frac{P}{n} \cdot \sum_{z=0}^{n} \frac{n}{n + P} \cdot \mathbb{P}(Z = z) = 2 \cdot \frac{P}{n} \cdot \frac{n}{n + P} \cdot \sum_{z=0}^{n} \mathbb{P}(Z = z) = \frac{2 \cdot P}{P + n}$$

It can be observed this can be attained when $\theta = 1$. Now that is identified that the highest F_1 score can be attained when everything is labelled with a positive label, the optimal expected F_1 score is determined. If we label everything with a positive label, then the recall becomes 1 and the precision becomes $\frac{P}{P+n}$. This results in a maximum attained F_1 score of $\frac{2 \cdot P}{P+n}$ when $\theta = 1$.

Baseline Hypergeometric

A second baseline, inspired by the top k selection of Joris Pries (N.D.), is randomly labelling d instances as positive and $n - d$ as negative. This approach implies that the number of positives is always $d = TP + FP$ and that $TP \sim h(n, P, d)$. From the hypergeometric distribution, it holds that $E(TP) = d\frac{P}{n}$. We can now calculate the expected *Recall*, *Precision* and F_1 score for this classifier.

$$\mathbb{E}(\text{Recall}) = \frac{\mathbb{E}(TP)}{P} = \frac{P \cdot d}{P \cdot n} = \frac{d}{n}$$

$$\mathbb{E}(\text{Precision}) = \mathbb{E}(\frac{TP}{TP + FP}) = \mathbb{E}(\frac{TP}{d}) = \frac{\mathbb{E}(TP)}{d} = \frac{d \cdot P}{d \cdot n} = \begin{cases} 0, & \text{if } d = 0 \\ \frac{P}{n}, & \text{else} \end{cases}$$

$$\mathbb{E}(F_1) = \mathbb{E}(\frac{2 \cdot TP}{TP + FP + P}) = \mathbb{E}(\frac{2 \cdot TP}{d + P}) = \frac{2 \cdot \mathbb{E}(TP)}{d + P} = \frac{2 \cdot d \cdot P}{n \cdot (d + P)}$$

Where it is clear how to optimize the expected recall and precision, this is not clear for the expected F_1 score. If the difference operator is strictly greater than 0, it holds that the expected F_1 score can be maximized by setting $d = n$. The forward difference operator is given by: $\Delta f : k \to f(k + 1) - f(k)$.

$$\Delta\mathbb{E}(F_1) = \frac{2\cdot(k+1)\cdot P}{n\cdot(k+1+P)} - \frac{2\cdot k\cdot P}{n\cdot(k+P)} = \frac{2\cdot P}{n}\left(\frac{k+1}{k+1+P} - \frac{k}{k+P}\right)$$

$$= \frac{2\cdot P}{n}\left(\frac{k^2+k+k\cdot P+P-k^2-k-P\cdot k}{(k+1+P)(k+P)}\right) = \frac{2\cdot P}{n}\left(\frac{P}{(k+1+P)(k+P)}\right) \geq 0$$

As the forward difference operator is always positive, it can be concluded that the function is increasing in d so d should be maximized. At most, $d = n$. This means that all instances are predicted with a positive label, which implies a maximum expected F_1 scores for the hypergeometric distribution is $\frac{2\cdot P}{n+P}$.

8.7.2 Multiclass Classification

Now, let us take a look at the multiclass classification problem. Suppose we have L classes to predict. This means that our confusion matrix M is of shape $(L \cdot L)$. Let us define M_{ij} as the number of observations with actual label i and predicted with label j. Table 8.3 shows how the

Table 8.3: Contingency matrix multiclass classification

Actual\Predicted	**1**	**2**	**...**	**L**	**Total**
1	M_{11}	M_{12}	...	M_{1L}	P_1
2	M_{21}	M_{22}	...	M_{2L}	P_2
...	...	...	...	...	...
L	M_{L1}	M_{L2}	...	M_{LL}	P_L

Now, for each label L, the positives in this case are all instances with actual label L, while the negative instances are all instances that do not have this label. From the contingency matrix, the True Positives for class i are the instances where the predictions for class i are indeed of label i.

$$TP_i = M_{ii}$$

The False Negatives for label i are all instances that have actual label i but do not have this prediction.

$$FN_i = P_i - TP_i = \sum_{j=1}^{c} M_{ij} - M_{ii}$$

For the False Positives, all instances with prediction i can be summed and the True Positives can be subtracted.

$$FP_i = \sum_{j=1}^{c} M_{ji} - M_{ii}$$

At last, the True Negatives in this case are the other instances that are not False Positive, False Negative or True Positive.

$$TN_i = N_i - FP_i = n - P_i - FP_i = n - \sum_{j=1}^{c} M_{ij} - \sum_{j=1}^{c} M_{ji} - M_{ii}$$

Now, suppose we want to determine the Recall, Precision and F1 score for each class separately, then the class specific TP, FP, FN and TN can be used.

Micro
As we have a multiclass model, it can be interesting to look at the performance of the model by combining the f1 scores of each class. Therefore, there are three approaches available for determining the model score: the micro score, the macro score and the weighted score. The following function shows how the micro score of the model can be determined:

$$Recall^{micro} = \frac{\sum_{c=1}^{C} TP_c}{\sum_{c=1}^{C} P_c} = \frac{\sum_{c=1}^{C} TP_c}{n}$$

Macro

The macro score is defined as:

$$metric^{macro} = \frac{1}{L} \sum_{l=1}^{L} metric_l$$

Weighted

The weighted score is defined as:

$$metric^{weighted} = \sum_{l=1}^{L} \frac{P_l}{n} metric_l$$

Chapter 9

Results

In this chapter, the results of the applied methods are discussed. In the first part, the results of the supervised learning techniques are shown. This section is used to determine whether attacks differ from normal behaviour. In the second part, the results of the anomaly detection techniques are given. At last, the results of detecting abnormal changes in the graphs are shown.

9.1 Supervised Learning

Before anomaly detection techniques can be applied, it is useful to check whether attacks indeed differ from normal traffic. Therefore, supervised learning techniques are applied in which the algorithms try to learn the characteristics of the different classes. Before the results of each dataset is discussed, the experimental setup is described which clarifies the steps taken in before applying the methods.

9.1.1 Experiment Setup

To acquire the results of the supervised learning algorithm, all datasets are randomly split in a train and a test set. For this research, a 80/20 train/test ratio with the holdout method is applied. As it is important to have a test set which is a representation of the train dataset, the test dataset is constructed using a stratified approach. This means that the class distributions between the train and test are approximately equivalent.

The datasets which are used in the supervised learning part should at least have one malicious instance and consist of at least 1000 instances. If there are no malicious classes, the supervised learning algorithm only has one class so there is no division to be learned. The 1000 instances requirement is selected as it is a convenient choice. This requirements imply that in the ISCX-IDS-2012 the SSL is ignored by only having normal instances and the SSL of the UNSW-NB15 dataset also as it only contains 182 instances.

The supervised learning algorithm are applied in both a multiclass fashion with all classes as well as a binary case. In the multiclass approach, the normal and each malicious class are used as input data, while in the binary approach only one class is considered in combination with normal traffic. The results in this section are the results of the model evaluation part. The following abbreviations are used for the models: Gaussian Naive Bayes (GBN), Decision Tree (DT), K-Neirest Neighbor algorithm with K neighbours considered (K-NN), Adaboost classifier with X estimators (ADA_X) and Random Forest Classifier with X estimators (RF_X). The binary baseline is determined by $\frac{2 \cdot P}{n+P}$, which is the optimal baseline for the binomial and hypergeometric classifier determined in Chapter 8.

9.1.2 CIC-IDS-2017

The supervised learning algorithms are first applied on the CIC-IDS-2017 protocol datasets. There are multiple attacks recorded for the protocols TCP, HTTP and SSH, while for the other considered protocols there is only one attack class. For the first three protocols, the multiclass approach is applied by training the mentioned algorithms on the training dataset and testing on the test set. Tables 9.1 and 9.2 show the macro and weighted F_1 scores of each classifier algorithm. Highlighted in bold are the scores of the best performing models for each protocol. It can be observed that the weighted F_1 baseline scores are higher compared to the macro F_1. The is caused by the largest class (normal traffic) which dominates the performance of the weighted F_1 score. Furthermore, it can be observed that the Gaussian Naive Bayes model does not necessarily perform better than the baseline. Other models perform relatively similar, but they all perform extremely better than the baseline. For SSH, it can be seen that the performance is the same for many different models. As it turns out, the models make exactly the same prediction for the observations and have therefore the same performance.

Table 9.1: CIC-IDS-2017 macro F_1 score

Model	**TCP**	**HTTP**	**SSH**
Baseline	0.06250	0.09091	0.33333
GNB	0.13445	0.11826	0.80454
1-NN	0.77049	0.94063	**0.96462**
DT	0.76089	0.97024	**0.96462**
RF_5	0.73704	**0.98029**	**0.96462**
RF_10	0.73777	0.97214	**0.96462**
RF_100	0.73726	0.97027	**0.96462**
ADA_5	0.77707	0.97043	**0.96462**
ADA_10	**0.77768**	0.96517	**0.96462**

Table 9.2: CIC-IDS-2017 weighted F_1 score

Model	**TCP**	**HTTP**	**SSH**
Baseline	0.32983	0.35438	0.52677
GNB	0.10671	0.20723	0.98255
1-NN	0.97387	0.99346	**0.99452**
DT	0.98560	0.99829	**0.99452**
RF_5	0.98587	**0.99844**	**0.99452**
RF_10	**0.98599**	0.99841	**0.99452**
RF_100	0.98588	0.99839	**0.99452**
ADA_5	0.98571	0.99835	**0.99452**
ADA_10	0.98446	0.99718	**0.99452**

Now that the multiclass F_1 scores are discussed, the performance of these classifiers can be evaluated by looking at the performance of each of the classes. Table 9.3 shows the binary F_1 for each class in the TCP dataset of the CIC-IDS-2017 dataset. The baseline scores in this table gives an indication on the relative occurrence of that class. The higher the baseline, the more samples in the dataset have that corresponding ground-truth label. So, as the normal class dominates the internet traffic, most connections are normal and thus the baseline is higher. Excluded from this performance table is the hearthbleed attack. As there is only one connection with this label, it is either in the training dataset or in the test data. Therefore, it is either trained but not tested or tested and not trained. The models outperform the baseline on most attacks. For some attacks the scores are even 1.0. The DDoS LOIC has for all models a 1.0 score and this indicates that the connections used during this attack are completely different from all other connections. For the FTP-Patator, only the decision tree classifier is able to make fully correct predictions. The other three models have classified one FTP-connection as normal traffic of 798 connections. A similar observation can be made with the SSH-Patator and Infiltration Cool Disk MAC where all classifiers are almost always able to predict these attacks correctly. For the Web Attack - SQL Injectionm, it can be observed that the nearest neighbour is the only model which can correctly predict these malicious activities while other models fail to make correct predictions. However, there are some attacks for which the classifiers are not able to make accurate predictions: Web Attack - Brute Force, Web Attack - XSS and In filtration Dropbox download. Eventhough the performance is higher than the baselines for these attacks, a higher F_1 score is desirable.

Table 9.3: CIC-IDS-2017 TCP F_1 score per attack best models

Class	Baseline	1-NN	DT	RF_100	ADA_10
BENIGN	0.66671	0.99016	0.99135	**0.9916**	0.99028
Botnet ARES	0.0092	0.95541	**0.95848**	**0.95848**	0.95947
DDos LOIC	0.16351	**1.0**	**1.0**	**1.0**	**1.0**
DoS Goldeneye	0.0143	0.97607	0.98032	**0.98446**	0.98508
DoS Hulk	0.26402	0.95708	0.98557	**0.98559**	0.98557
DoS SlowHTTPtest	0.02946	0.33932	0.74683	**0.75069**	0.74951
DoS Slowloris	0.00723	0.74208	0.74425	**0.75662**	0.74437
FTP-Patator	0.0074	0.99937	**1.0**	0.99937	0.99937
Infiltration Cool disk MAC	5e-05	**1.0**	**1.0**	**1.0**	**1.0**
Infiltration Dropbox download	**4e-05**	0.0	0.0	0.0	0.0
Portscan	0.36064	0.99107	0.99403	**0.99413**	0.99094
SSH-Patator	0.00553	0.99749	**1.0**	**1.0**	**1.0**
Web Attack - Brute Force	0.00254	0.26343	0.26435	**0.26527**	0.26178
Web Attack - SQL Injection	2e-05	**1.0**	0.4	0.0	0.66667
Web Attack - XSS	0.00126	**0.11636**	0.10909	0.10989	0.10989

As some of the F_1 performances of the classifiers are not satisfiable for some attacks, the feature importances of a model could explain why the performance is far from optimal. Figure 9.1 show the feature importance of the decision tree per class for the TCP protocol. These feature importances are created by creating one decision tree for each class where only the connections of that class are used and the normal traffic to train the model. It can be observed that indeed the FTP-Patator requires service FTP and the SSH-Patator uses SSH services, which is not surprising. What is unexpected is that the Web Attacks heavily depend on the duration of a connection, which is problematic when normal connections have similar duration as these Web attacks.

Figure 9.1: CIC-IDS-2017 TCP feature importance decision tree

While most attacks are distinguishable from other attacks and normal traffic on TCP layer, there were some attacks for which it is not the case. Maybe the features on this layer are not explanatory for these attacks. Looking also at features on application layer might help finding these attacks. In the labelling scheme of the CIC-IDS-2017 dataset can be observed that most attacks in the CIC-IDS-2017 dataset use the HTTP protocol. Therefore, the considered classifiers are applied on the HTTP layer dataset. Table 9.4 shows the F_1 score performance of the classifiers on this dataset. It can be observed that the performance on this dataset is almost perfect on this dataset. Attacks which could not be distinguished on the TCP layer can be distinguished on HTTP level. The only attack which is hard to detect is the Portscan attack. However, portscans target multiple ports so it is not expected that this attack is identifiable only looking on the HTTP layer features.

Table 9.4: CICIDS2012 HTTP F_1 score per attack best models

Class	**Baseline**	**1-NN**	**DT**	**RF_10**	**ADA_10**
BENIGN	0.6518	0.99967	0.99971	**0.9998**	0.99976
Botnet ARES	0.00275	0.9932	**1.0**	**1.0**	**1.0**
DDos LOIC	0.30389	0.98406	**0.99688**	**0.99688**	**0.99688**
DoS Goldeneye	0.02919	0.98601	0.99271	**0.99428**	0.95485
DoS Hulk	0.45777	0.98987	0.99791	**0.99799**	0.9959
DoS SlowHTTPtest	0.00529	**0.98936**	0.98932	0.98932	0.99109
DoS Slowloris	0.00837	0.99001	**0.99443**	**0.99443**	**0.99443**
Portscan	0.00077	**0.76923**	0.71875	0.73529	0.69841
Web Attack - Brute Force	0.02701	0.99387	0.99386	**0.99523**	0.99522
Web Attack - SQL Injection	4e-05	0.66667	**1.0**	**1.0**	**1.0**
Web Attack - XSS	0.00681	0.98503	0.98904	**0.99037**	**0.99037**

Similar to the TCP layer, the feature importances can be investigated to determine which features influence the predictions of the observations. The features importances of the decision tree trained on the training dataset of the HTTP can be observed in Figure 9.2.

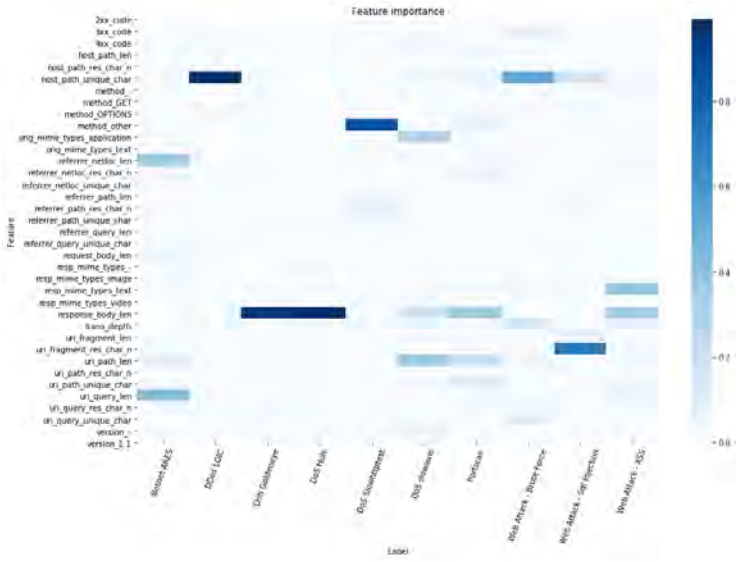


Figure 9.2: CIC-IDS-2017 HTTP feature importance decision tree

The D(D)oS attacks are heavily influenced by single features. For the DoS SlowHTTPtest, the feature *method_other* has a feature importance of 0.86. As it turns out, the DoS SlowHTTPtest uses a method called "FAKEVERB" which almost uniquely identifies the the SlowHTTPtest attack.

For the DoS Goldeneye and the DoS Hulk the *response_body_len* turns out to be extremely important. A value of 11321 is used 95.5% of all training instances for the DoS Goldeneye and 99.5% for all training instances for the DoS Hulk. For the DDoS LOIC the feature *host_path_unique_char* only consisted of 1 character namely the dash symbol.

The SSH protocol dataset consists of normal traffic and two attacks: port scans and the SSH-Patator. For the SSH-Patator attack in the SSH dataset, the F_1 scores for each classifier is above 0.97. The baseline for this attack here is 0.533. The other attack in the SSH dataset is the portscan attack. This attack is not only recorded for the SSH protocol but also the only attack in the DNS, UDP and FTP protocols. As this attack is recorded over multiple datasets, the classifiers are applied on all protocols of the CIC-IDS-2017 datasets to investigate in which this portscan can best be detected. To make fair comparisons, only the normal traffic and the portscans connections are used in a binary classification problem. Table 9.5 shows the F_1 scores of some classifiers applied on all datasets. On the TCP layer, the F_1 score is highest, which is expected as portscans are not targeted on one single port. Portscans are however not detectable on the FTP protocol.

Table 9.5: F_1 scores binary classification port scan attack vs normal

Model	**TCP**	**UDP**	**HTTP**	**DNS**	**FTP**	**SSH**
Baseline	0.46805	0.00097	0.00159	0.0002	**0.00803**	0.02985
1-NN	0.99072	0.96154	0.88889	0.89474	0.0	**0.90323**
DT	0.99404	0.94175	0.85714	**0.9**	0.0	**0.90323**
RF_100	**0.99413**	**0.96117**	**0.92105**	0.87805	0.0	**0.90323**
ADA_5	0.99403	0.96078	0.89744	**0.9**	0.0	**0.90323**

9.1.3 ISCX-IDS-2012

The second dataset which is used in this research is the ISCX-IDS-2012. In the datasets of the ISCX-IDS-2012 experiment, there are only two protocols for which multiple attack types are recorded in the datasets: TCP and SSH. These protocols are used for the multiclass classification problem to determine whether the attacks which occur in the dataset are indeed different from normal traffic. Tables 9.6 and 9.7 show the macro and weighted F_1 performance of the different classifiers on the test-dataset of these protocols. Interesting to note in the performances of the classifiers on the SSH protocol is that the tree based classifiers perform equivalently for the macro and the weighted case. In the SSH protocol, there are connections with the actual label "unknown" as discussed in chapter 5. As none of the 18 connections with actual label "unknown" are predicted with this label in all models, the F_1 score of this class is 0. As the F_1 score of this class is 0, the macro F_1 score dramatically decreases as there are only 4 classes in the SSH protocol. There is also one connection with actual label exfiltration and this is also incorrectly predicted.

Table 9.6: ISCX-IDS-2012 macro F_1 score

Model	**TCP**	**SSH**
Baseline	0.09091	0.25
GNB	0.17818	0.40119
1-NN	0.74804	0.30429
DT	0.78108	**0.45952**
RF_5	0.68949	**0.45952**
RF_10	0.78035	**0.45952**
RF_100	0.77749	**0.45952**
ADA_5	**0.78415**	**0.45952**
ADA_10	0.78382	**0.45952**

Table 9.7: ISCX-IDS-2012 weighted F_1 score

Model	**TCP**	**SSH**
Baseline	0.92719	0.50567
GNB	0.0186	0.77602
1-NN	0.99251	0.63345
DT	0.99274	**0.91339**
RF_5	0.99295	**0.91339**
RF_10	**0.99298**	**0.91339**
RF_100	0.99295	**0.91339**
ADA_5	0.99282	**0.91339**
ADA_10	0.99272	**0.91339**

As the previous two tables only show the general performance of the classifiers over all classes, Table 9.8 show the F_1 performance for the models per class. As can be observed, the backdoor and unknown attacks are outperforming the baseline, but are still not accurately predicted correctly in most cases. Similar to the CIC-IDS-2017, the DDoS Slowloris attack in this dataset has a F_1 score which is of the same performance level. The same can be said about the performance of the SSH Brute Force attack, which has similarities with the SSH-Patator. The performance of the classifiers on the portscan class is lesser than the performance in the CIC-IDS-2017, but there are less connections in this dataset with this label.

Table 9.8: ISCX-IDS-2012 TCP F_1 score per class

Class	**Baseline**	**1-NN**	**DT**	**RF_100**	**ADA_5**
BENIGN	0.98096	0.99607	0.99619	**0.99629**	0.99623
Backdoor	0.00095	**0.30612**	0.28772	0.2973	0.28671
DDoS Botnet	0.03854	0.95333	0.95562	**0.96052**	0.95748
DDoS Slowloris	0.00648	0.79277	**0.7926**	0.79405	0.79405
Exfiltration	0.0	0.66667	**1.0**	**1.0**	**1.0**
Exploit	5e-05	0.66667	0.69565	0.64	**0.72727**
Portscan	0.02265	0.84885	0.85564	**0.85569**	0.85543
SQL Password	3e-05	**0.66667**	**0.66667**	**0.66667**	**0.66667**
SSH Brute Force	0.00488	0.99798	**0.99899**	**0.99899**	**0.99899**
Unknown	9e-05	0.33333	**0.34286**	**0.34286**	**0.34286**
Vulnerable SMB	0.0	**1.0**	**1.0**	**1.0**	**1.0**

In the ISCX-IDS-2012, the port scan attacks are only recorded on the TCP, UDP and DNS protocol. Table 9.9 shows the F_1 scores of the binary classification problem on datasets with only normal traffic and the port scan attacks. The TCP protocol is the only protocol where these kinds of attacks can be accurately detected. Table 9.10 show that the performance of the classifiers for the DDoS Botnet is similar between TCP and HTTP.

Table 9.9: ISCX-IDS-2012 F_1 score portscan

Model	**TCP**	**UDP**	**DNS**
Baseline	0.02325	3e-05	2e-05
GNB	0.03175	5e-05	**0.00012**
KNN_1	0.67871	0.0	0.0
DT	0.85521	0.0	0.0
RF_5	0.85564	0.0	0.0
RF_10	0.85564	0.0	0.0
RF_100	0.85569	0.0	0.0
ADA_5	0.85561	**0.5**	0.0
ADA_10	**0.85583**	**0.5**	0.0

Table 9.10: ISCX-IDS-2012 F_1 score DDoS Botnet

Model	**TCP**	**HTTP**
Baseline	0.03922	0.62764
GNB	0.58635	0.80198
KNN_1	0.9534	N.F.[a]
DT	0.95553	**0.9556**
RF_5	0.95864	**0.9556**
RF_10	0.95926	**0.9556**
RF_100	**0.96038**	**0.9556**
ADA_5	0.95657	0.95505
ADA_10	0.95678	0.95535

[a] N.F. = Not feasible. As the train dataset contains around 7.2 million connections and the test dataset 1.8 million, the total number of comparisons is $12.96 \cdot 10^{12}$.

9.1.4 UNSW-NB15

The last dataset for which the supervised learning classifiers are applied is the UNSW-NB15. The classifiers are applied on all protocols except the SSL dataset. Table 9.11 and Table 9.12 show the macro and weighted F_1 scores for the multiclass classification problems. As each protocol contains multiple classes, each dataset requires a multiclass classification algorithm. It can be observed in the macro F_1 scores that the classifiers outperform the baselines. The interesting part is that the GBN model is the best classifier on the FTP protocol. In other cases the GNB model is outperformed by the other models.

Table 9.11: UNSW-NB15 macro F_1 score

Model	**TCP**	**UDP**	**HTTP**	**DNS**	**FTP**	**SSH**
Baseline	0.1	0.11111	0.11111	0.16667	0.2	0.33333
GNB	0.11923	0.34653	0.20262	0.04464	**0.74121**	0.70764
KNN_1	0.631	0.70029	0.75081	0.80111	0.57373	**0.79995**
DT	0.60598	0.69511	0.77838	0.86671	0.67716	**0.79995**
RF_5	0.64567	0.6896	0.78005	0.87543	0.70897	**0.79995**
RF_10	0.63618	0.70558	**0.79104**	0.88638	0.70897	**0.79995**
RF_100	**0.64798**	**0.71302**	0.78609	**0.88844**	0.67716	**0.79995**
ADA_5	0.62701	0.67308	0.78358	0.83015	0.67716	**0.79995**
ADA_10	0.62739	0.67031	0.78461	0.82505	0.67716	**0.79995**

The weighted F_1 scores of the baselines are important to note in Table 9.12. As can be seen in Chapter 7, the attack classes instances combined are very rare. The normal traffic is very dominant in the datasets of the UNSW-NB15. As a result, in the weighted F_1 score is primarily influenced by the dominant class in the dataset, which is in most cases normal traffic. As a result, the baselines of the protocols are very close to one. Therefore, a weighted F_1 score of almost 1 cannot be interpret as good without looking at the baselines of the datasets. Still, the classifiers, except the GNB, do outperform the baseline is almost all scenarios.

Table 9.12: UNSW-NB15 weighted F_1 score

Model	**TCP**	**UDP**	**HTTP**	**DNS**	**FTP**	**SSH**
Baseline	0.92953	0.95451	0.87192	0.91638	0.94761	0.99921
GNB	0.60578	0.97771	0.13004	0.00037	0.99471	0.99766
KNN_1	0.98325	0.98643	0.98449	0.99574	0.99109	**0.99967**
DT	0.98321	0.98622	0.9867	0.99846	0.99478	**0.99967**
RF_5	0.98486	0.98645	0.98815	0.99855	**0.99486**	**0.99967**
RF_10	0.98504	0.98667	0.98887	**0.99859**	**0.99486**	**0.99967**
RF_100	**0.98547**	**0.98669**	**0.98909**	**0.99859**	0.99478	**0.99967**
ADA_5	0.98363	0.98606	0.98769	0.99839	0.99478	**0.99967**
ADA_10	0.98375	0.986	0.98804	0.99841	0.99478	**0.99967**

As most attacks occur in more than one protocol layer, it is interesting to see on which protocol layer attacks can best be detected. Table 9.13 shows the F_1 score for each attack on each protocol layer for the random forest model with 100 estimators. Missing values here means that no F_1 score could be computed because the attacks did not occur on that layer. Interesting to see here is that there is not one specific protocol for which all attacks can be detected but that the highest scores can be detected on several protocols. The only attack which is not detected is the analysis attack.

Table 9.13: RF_100 F_1 protocol performance

Class	TCP	UDP	HTTP	DNS	FTP	SSH
Analysis	**0.09174**	-	0.01818	-	-	-
BENIGN	0.99432	0.99489	0.99665	0.99934	0.99724	**0.99985**
Backdoor	0.66667	**1.0**	0.94382	-	-	-
DoS	0.5	0.56376	0.5976	0.69231	**0.72727**	-
Exploits	0.85754	0.61157	0.93484	0.77707	**0.9484**	0.4
Fuzzers	0.59453	0.35532	0.75039	**0.98177**	0.71287	-
Generic	0.87901	0.6683	0.95788	0.99125	0.0	**1.0**
Reconnaissance	0.81802	**0.97065**	0.92627	0.88889	-	-
Shellcode	0.46254	**0.58599**	-	-	-	-
Worms	0.61538	0.66667	**0.94915**	-	-	-

9.2 Anomaly Detection

The second section of this chapter consists of the results of applying unsupervised learning algorithms on the datasets. These algorithms do not use the labels as opposed to the supervised learning algorithms. In the first part of this section, the experimental setup is given which describes how the datasets are prepared before applying anomaly detection techniques. Furthermore, it is shown how the anomaly scores are determined for each instance. Afterwards, the results of the PCA and Isolation Forest methods are discussed for each dataset.

9.2.1 Experiment Setup

While class balance is useful for supervised learning techniques, class imbalance is occurring in anomaly detection datasets. When looking at the intrusion ratio in Table 7.2, there are some protocols for which the intrusion ratio is relatively large when considering malicious connections as anomalies. Therefore, for the protocols in which more than 5% of the connections are malicious, the malicious connections are downsampled with the restriction that there is at least one connection of each attack kind. This downsampling results in the following intrusion ratio, which can be see in Table 9.14. The 5% values indicate that these protocols are downsampled, which thus involves five protocols over the tree datasets.

Table 9.14: Percentage Malicious Activity Anomaly Detection

Protocol	CIC-IDS-2017	ISCX-IDS-2012	UNSW-NB15
TCP	5.00%	3.74%	3.61%
UDP	0.05%	0.0012%	2.31%
DNS	0.01%	0.0009%	4.34%
HTTP	5.00%	5.00%	5.00%
SSL	0.0003%	0%	-
SSH	5.00%	5.00%	0.04%
FTP	0.32%	0.24%	2.68%

To evaluate the anomaly detection techniques, the 10-fold cross validation technique is applied on the datasets. This implies that each instances gets one anomaly score. This approach is performed five times with different folds so that each instance in the end has five anomaly scores. Th end anomaly score of an instance is calculated by taking the mean of these five anomaly scores. With this approach, the anomaly scores are not dependent on only one model but multiple to get an good estimate of the anomaly score of the instance. The following abbreviations are used for the anomaly detection models: PCA with X% of the principle components with the highest explained variance (PCA_X) and isolation forest with X estimators (IForest_X).

9.2.2 CIC-IDS-2017

The first datasets for which the anomaly detection algorithms are applied is the CIC-IDS-2017 dataset. Figure 9.3 shows the progress of the evaluation metrics for the PCA model using 50% of the principle components applied on the TCP dataset. Each instance obtains an anomaly score and this list of instances are descending ordered by their anomaly scores. The threshold level (θ) determines that X% of the highest anomaly scores are labelled as malicious while the lowest 1-X% is labelled as normal. By increasing this threshold, the evaluation metrics change. There are several interesting moments which are interesting to evaluate. First, what is the F_1 score when the anomaly ratio is the threshold. At this point, the recall is equal to the precision, which is thus means that it is the F_1 score. Secondly, what is the the maximum attained F_1 score on the anomaly scores. At last, it might be interesting to see at what threshold level all anomalies are predicted with the malicious label. This indicates at which level all traffic below the threshold is normal.

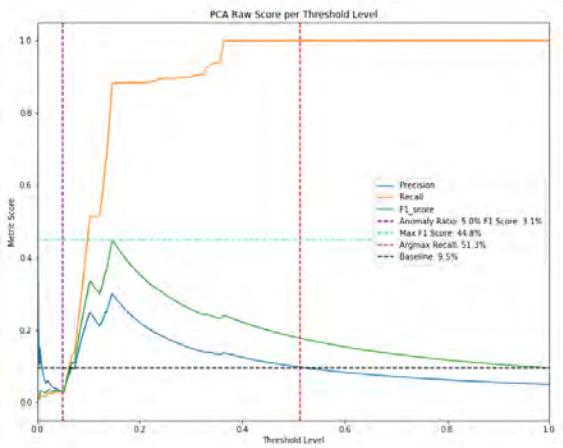


Figure 9.3: Evaluation metrics per theshold level CIC-IDS-2017 TCP PCA 0.5

Anomaly ratio malicious threshold

Table 9.15 below gives the F_1 scores for the considered models when the threshold equals the intrusion ratio. It can be observed that only the models applied on the SSH and UDP protocols outperform the baseline in some cases. For the other protocols the performance is 0 as all malicious labelled traffic is actually normal. On TCP, only the isolation forest with 5 estimators outperformed the baseline. The models performed on the SSH protocol actually worked great comparing the results with the baseline.

Table 9.15: CIC-IDS-2017 F_1 score when $\theta = \frac{P}{n}$

Model	TCP	UDP	DNS	HTTP	FTP	SSH	SSL
Baseline	**0.09524**	0.00097	**0.0002**	**0.09523**	**0.00644**	0.09506	**1e-05**
IForest_5	0.00396	0.0	0.0	0.00486	0.0	0.53846	0.0
IForest_10	0.01209	0.0	0.0	0.00383	0.0	0.53846	0.0
PCA_0.1	0.03125	**0.00397**	0.0	0.02061	0.0	**0.8022**	0.0
PCA_0.5	0.03079	**0.00397**	0.0	0.02061	0.0	**0.8022**	0.0
PCA_0.9	0.03079	**0.00397**	0.0	0.02061	0.0	**0.8022**	0.0

The F_1 score in Table 9.15 show the preformance of the models by comparing normal and a combination of all attack classes. It might be interesting to see which attacks in this set of attack classes were detected using this threshold level. Figure 9.4 shows in percentage what part of

which class is predicted as malicious and which is predicted normal given the intrusion ratio as threshold level for the isolation forest model with 5 estimators. It can be observe that using this threshold level results in being able to detect the heartbleed attack. While in the supervised learning techniques it was not possible to detect this attack as there is only one instance, this model is able to detect this attack using this threshold level. For many other attacks only a small part of the attacks are detected for this protocol.

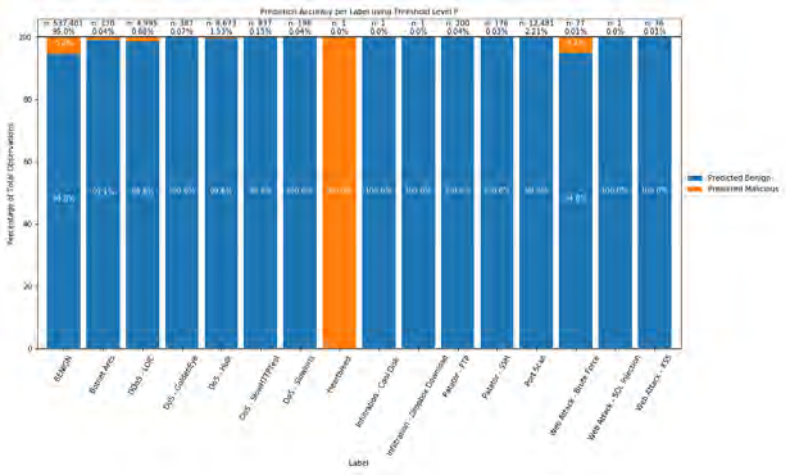


Figure 9.4: CIC-IDS-2017 TCP IForest_5 normal and malicious prediction per class

Figure 9.5 shows the boxplots of the anomaly scores for each class. It can be observed that the heartbleed attack is in the top of the anomaly scores. Attack such as the Web Attack Brute Force, the FTP and SSH patator have relatively lower anomaly scores compared to other malicious classes. This explains why some of the attacks are not detected seen in Figure 9.15.

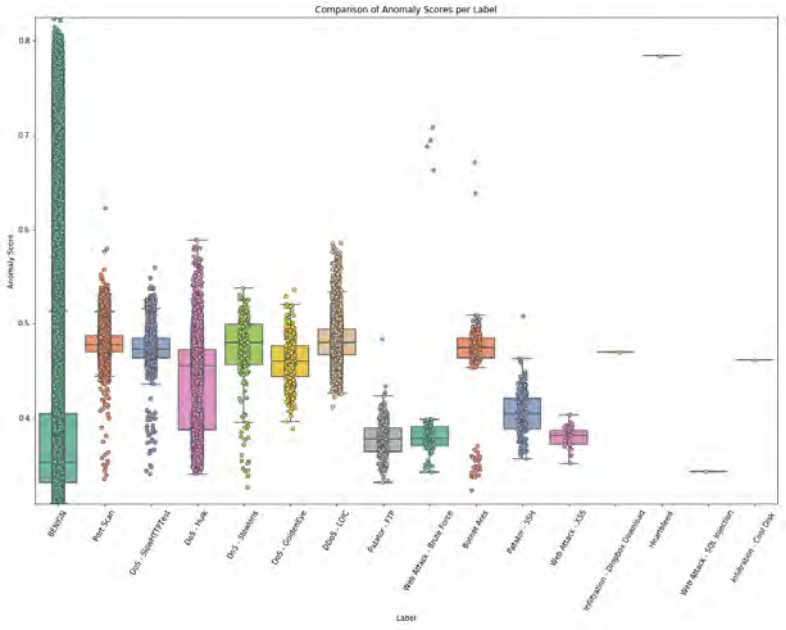


Figure 9.5: CIC-IDS-2017 TCP IForest_5 anomaly scores boxplot

Highest obtained F_1 score

Instead of looking at the F_1 score at the anomaly ratio threshold, it can be interesting to see what the highest attainable performance is. Table 9.16 states the highest attained F_1 scores. This is, of course, a very optimal score as the user has to know the threshold for which this F_1 score can be obtained.

Table 9.16: CIC-IDS-2017 highest obtained F_1 score

Model	TCP	UDP	DNS	HTTP	FTP	SSH	SSL
Baseline	0.09524	0.00097	0.0002	0.09523	0.00644	0.09506	1e-05
IForest_5	0.36796	0.02643	0.00117	0.09569	0.00665	0.81073	0.00042
IForest_10	0.3576	0.03537	0.00115	0.09609	0.00681	0.81073	0.00041
PCA_0.1	0.45154	**0.07227**	0.00548	**0.16293**	**0.01794**	**0.90084**	0.00016
PCA_0.5	0.4519	0.05664	**0.00599**	0.12201	**0.01794**	**0.90084**	0.00017
PCA_0.9	**0.45347**	0.06185	0.00547	0.12274	0.01072	0.89562	0.00016

Similar to the threshold level anomaly ratio, the attacks self can be evaluated on whether indeed the attacks are found when using the maximum F_1 score anomaly detection threshold level. Figure 9.6 shows the results at this threshold level. As the threshold level is higher than the previous case, more benign traffic is labelled as malicious. Therefore, the recall has increased. For all the other attacks we can see a significant increase in anomaly detection accuracy. The attacks which are hard to detect on the TCP supervised learning techniques are the web attacks. Similar here, they cannot be found in the TCP dataset.

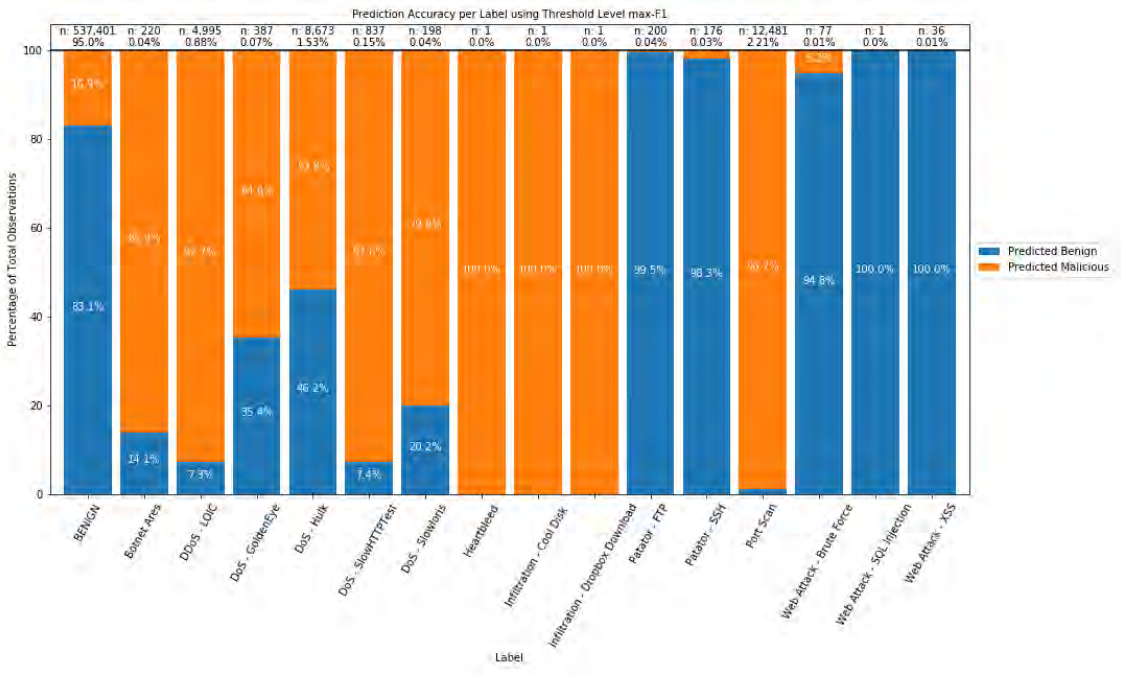


Figure 9.6: CIC-IDS-2017 TCP IForest_5 highest attained F_1 performance per class

Optimal recall threshold

The third performance indicator is the threshold level for which the recall is optimal. This indicator shows the threshold level for which the recall is 100%, or TP = P. Table 9.17 shows the threshold levels for which the the recall level is 100% for the five different models. So, in the TCP protocol for the PCA model with 50% components used all malicious connections are in the highest 50% anomaly scores. For the HTTP protocol, it turns out that the median of the DoS Hulk attack is around 0.38 for Isolation Forest model with 10 estimators and the median BENIGN is 0.41. The DoS Hulk consists of 7,882 observations, which are almost all connections.

Table 9.17: CIC-IDS-2017 optimal recall threshold

Model	**TCP**	**UDP**	**DNS**	**HTTP**	**FTP**	**SSH**	**SSL**
Anomaly Ratio	5.0%	0.048%	0.01%	5.0%	0.323%	4.99%	≈0.0%
IForest_5	83.316%	65.512%	39.841%	99.994%	98.667%	73.149%	**1.434%**
IForest_10	66.549%	53.632%	19.153%	99.936%	94.548%	**44.544%**	1.464%
PCA_0.1	53.978%	**28.005%**	**4.042%**	**93.924%**	96.769%	91.939%	3.737%
PCA_0.5	**50.879%**	96.128%	4.505%	96.623%	96.769%	96.582%	3.636%
PCA_0.9	53.241%	48.177%	4.306%	97.246%	**88.651%**	64.504%	3.649%

9.2.3 ISCX-IDS-2012

The second experiment for which the unsupervised learning techniques are applied is the ISCX-IDS-2012. Similar to the CIC-IDS-2017 experiment, the performance of the models are measured at the F_1 score at the anomaly ratio, the maximum performance and the moment when the recall is optimal.

Threshold anomaly ratio

Table 9.18 shows the performance of the models when the intrusion ratio is used as threshold level. For the UDP, DNS and FTP protocols, the models perform worse than the baseline. This means that all connections which are labelled with the malicious label are actually benign and therefore the F_1 score is 0. For the TCP and the SSH protocol the baselines are outperformed. On the TCP layer, the PCA model outperforms the IForest model, while for the SSH protocol it is the other way around.

Table 9.18: ISCX-IDS-2012 F_1 score when $\theta = \frac{P}{n}$

Model	**TCP**	**UDP**	**DNS**	**HTTP**	**FTP**	**SSH**
Baseline	0.07204	**2e-05**	**2e-05**	**0.09524**	**0.00477**	0.09514
IForest_5	0.00937	0.0	0.0	4e-05	0.0	**0.25069**
IForest_10	0.08342	0.0	0.0	4e-05	0.0	**0.25069**
PCA_0.1	**0.20554**	0.0	0.0	0.01188	0.0	0.00275
PCA_0.5	0.20492	0.0	0.0	0.01184	0.0	0.00275
PCA_0.9	0.20515	0.0	0.0	0.01187	0.0	0.0

Similar to the CIC-IDS-2017 dataset, it is interesting to see which attacks are detected using this threshold. Figure 9.7 shows the percentages being predicted with an anomalous label for the upper threshold of 3.73%, which is the intrusion ratio. As can be observed, the exfiltration and infiltration connections are actually predicted with the correct label, while almost all unknown connections are also predicted correctly. The vulnerable SMB attack however is not detected at all when using this threshold.

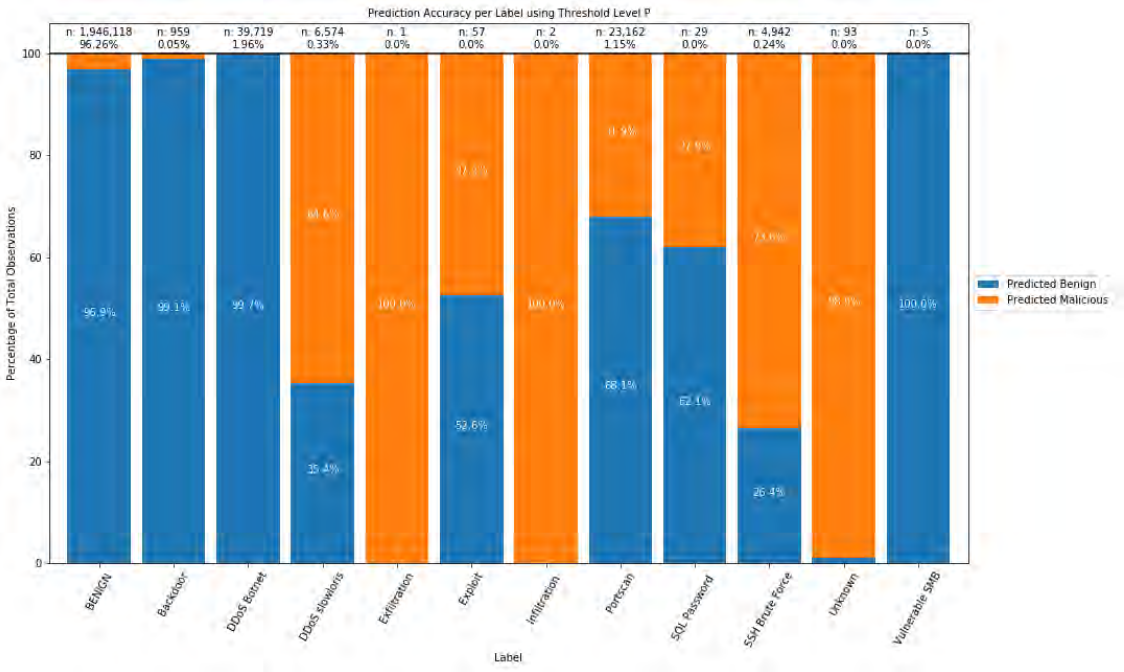


Figure 9.7: ISCX-IDS-2012 TCP PCA_0.1 performance per class

Highest F_1 score

Table 9.19 shows the performance of the models when looking at the highest attained F_1 score. These are optimistic in the sense that the user should know that the threshold should be set at this level. In comparison with the previous scores, the UDP, DNS and FTP gain little by changing the threshold level. The performance of the TCP and SSH layer have increase by almost 50% for both protocols.

Table 9.19: ISCX-IDS-2012 Highest attained F_1 score per model

Model	**TCP**	**UDP**	**DNS**	**HTTP**	**FTP**	**SSH**
Baseline	0.07204	2e-05	2e-05	0.09524	0.00477	0.09514
IForest_5	0.33199	0.0002	0.0005	0.11497	0.08511	**0.4454**
IForest_10	**0.34683**	0.00033	0.00159	0.11164	0.05839	**0.4454**
PCA_0.1	0.2838	**0.00108**	**0.00193**	0.14994	**0.19512**	0.0952
PCA_0.5	0.28277	**0.00108**	0.00192	**0.15944**	**0.19512**	0.0952
PCA_0.9	0.28216	**0.00108**	0.00192	0.15722	0.00836	0.0952

Figure 9.8 on the next page shows the anomaly normal prediction per class when the threshold is the highest attained F_1 score. In contrast with the previous figure, almost all attacks are completely detected. However, 14% of the normal traffic is predicted malicious to attain this F_1 score of 0.34.

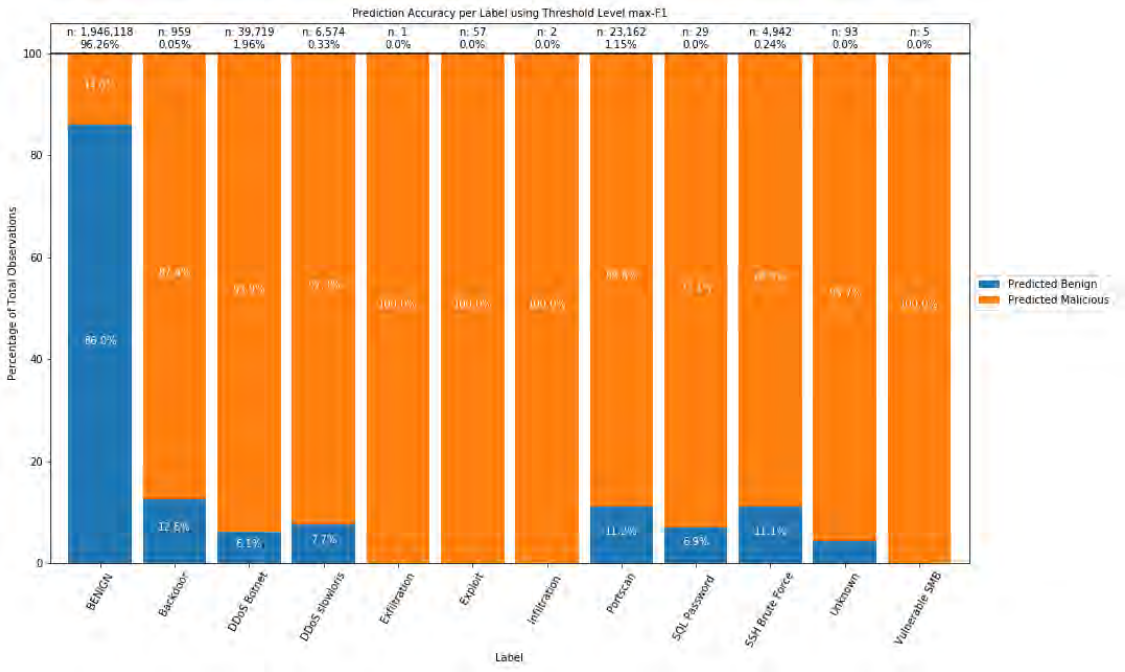


Figure 9.8: ISCX-IDS-2012 TCP IForest_5 anomaly normal prediction per class

Optimal recall threshold

In this last part of this dataset, the optimal recall threshold is determined for each dataset. Table 9.20 shows the threshold levels for which the recall is 1.0. It can be observed that for the TCP and HTTP protocols there are some anomalous connections with low anomaly scores, which imply that the threshold should be high to be able to detect these connections. In contrast, the DNS and FTP quickly have a recall of 1.0 at a low threshold level of 0.948 and 2.212 respectively for the PCA_0.5 model.

Table 9.20: ISCX-IDS-2012 optimal recall threshold per dataset

Model	**TCP**	**UDP**	**DNS**	**HTTP**	**FTP**	**SSH**
Anomaly Ratio	3.737%	0.001%	0.001%	5.0%	0.239%	4.994%
IForest_10	70.826%	28.591%	1.146%	99.994%	7.95%	71.67%
IForest_5	**89.64%**	**23.918%**	3.643%	99.996%	5.38%	**49.34%**
PCA_0.1	99.731%	63.873%	0.945%	99.881%	**2.212%**	99.931%
PCA_0.5	99.993%	68.041%	**0.948%**	**98.726%**	**2.212%**	99.931%
PCA_0.9	99.243%	65.626%	**0.948%**	99.414%	90.855%	99.931%

9.2.4 UNSW-NB15

The last dataset which is used for the anomaly detection algorithm is the UNSW-NB15. For this dataset, only the HTTP protocol is downsampled. Again, like previous datasets, the F_1 score at the intrusion ratio is calculated, the highest attained F_1 score and the optimal recall point is determined.

Threshold anomaly ratio

Table 9.21 shows the results of the models on the UNSW-NB15 dataset using the thresholdlevel $\theta = \frac{P}{n}$. It can be observed that the models perform much better on the HTTP, FTP and UDP protocols compared to the previous datasets. The PCA model outperforms the isolation forest models in almost all cases.

Table 9.21: UNSW-NB15 threshold anomaly ratio F_1 score

Model	TCP	UDP	DNS	HTTP	FTP	SSH
Baseline	**0.06966**	0.04516	0.0831	0.09524	0.05221	0.00079
IForest_5	0.05804	0.23293	**0.2086**	0.37467	0.5896	**0.1**
IForest_10	0.04005	0.31437	0.20484	0.53509	0.5736	0.05
PCA_0.1	0.06601	0.45591	0.19687	0.52867	0.6272	0.05
PCA_0.5	0.06583	**0.48933**	0.19863	**0.54561**	0.6384	0.05
PCA_0.9	0.06586	0.46685	0.1959	0.54404	**0.6392**	0.05

Figure 9.9 shows the performance of the model on the FTP application layer. It can be seen that the DoS, Fuzzers and Generic are totally found when using the anomaly ratio as anomaly score. Only the exploits are not entiry obtained. As 45% of 1,036 are not labelled as malicious, this indicates that this number is the amount of connections which are labelled as malicious but are actually normal traffic.

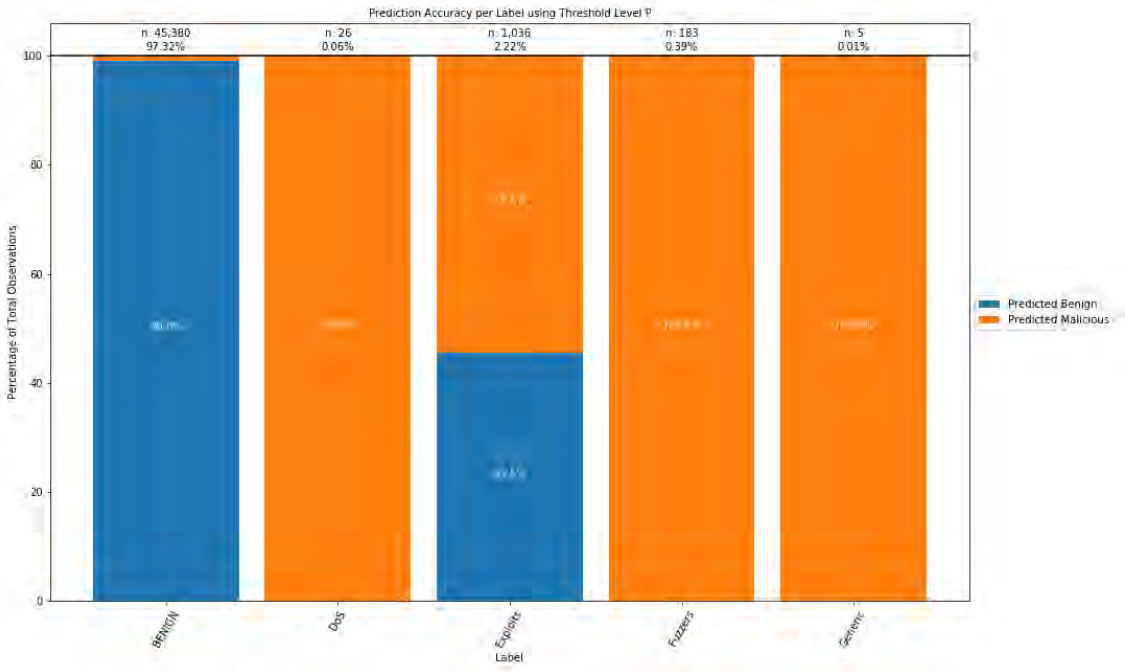


Figure 9.9: UNSW-NB15 FTP PCA_0.9 performance at anomaly ratio threshold

When taking a loot at the HTTP performance, it can be observed that again the fuzzers are almost all detected. Also the backdoor and worm attacks are identified. However, the analysis attack is barely detected.

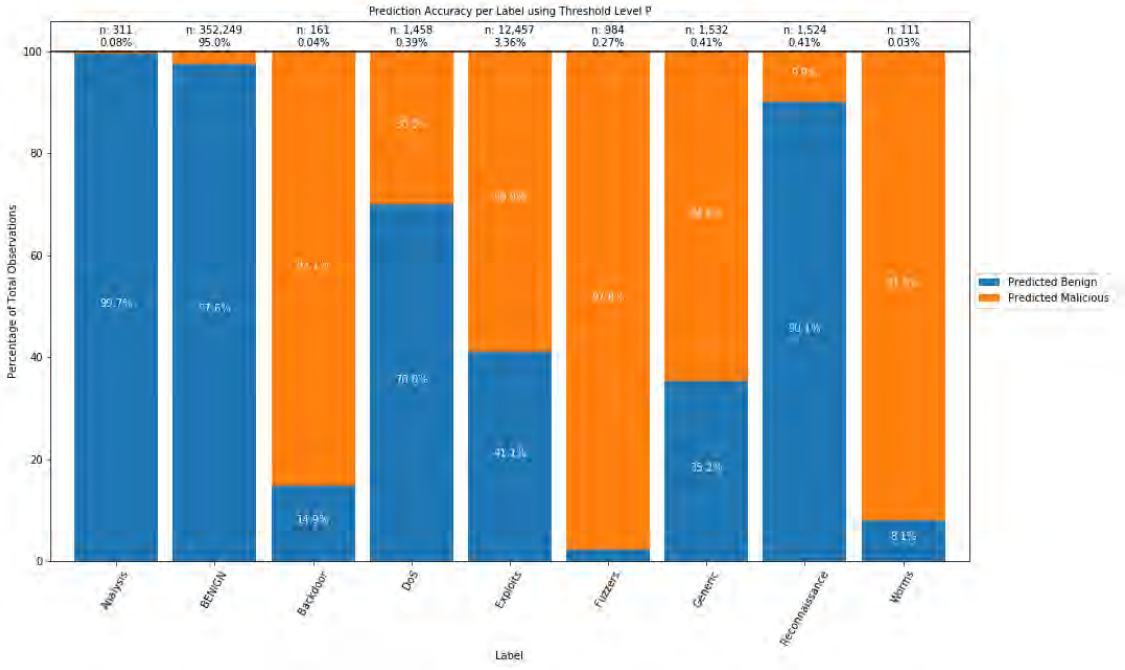


Figure 9.10: UNSW-NB15 HTTP PCA_0.5 performance at anomaly ratio threshold

Highest F_1 score

In the next situation, the highest attained F_1 scores are analysed. Table 9.22 shows these F_1 scores for the UNSW-NB15 dataset. It can be observed that the performance of the UDP and FTP dataset is reasonable high. In contrast to previous datasets, the models do not perform great on the SSH protocol.

Table 9.22: UNSW-NB15 highest F_1 score

Model	TCP	UDP	DNS	HTTP	FTP	SSH
Baseline	0.06966	0.04516	0.0831	0.09524	0.05221	0.00079
IForest_5	0.12235	0.36282	0.6179	0.43925	0.76666	0.11321
IForest_10	0.11804	0.47286	0.63272	0.54095	0.76923	0.14035
PCA_0.1	**0.15911**	0.73387	0.70637	0.54415	0.82243	**0.17167**
PCA_0.5	0.14223	0.73662	**0.70638**	**0.5637**	**0.8315**	**0.17167**
PCA_0.9	0.13317	**0.73677**	0.70558	0.55152	0.82763	**0.17167**

When looking at the performance of the models on the DNS protocols between Table 9.21 and Table 9.22, there is an great difference in F_1 score. When looking at the normal malicious predictions in Figure 9.11 at the threshold level with the highest attained F_1 score, it can be observed that almost all attacks are detected, except for some DoS connections. Only 3.8% of the normal traffic is labelled malicious in this situation.

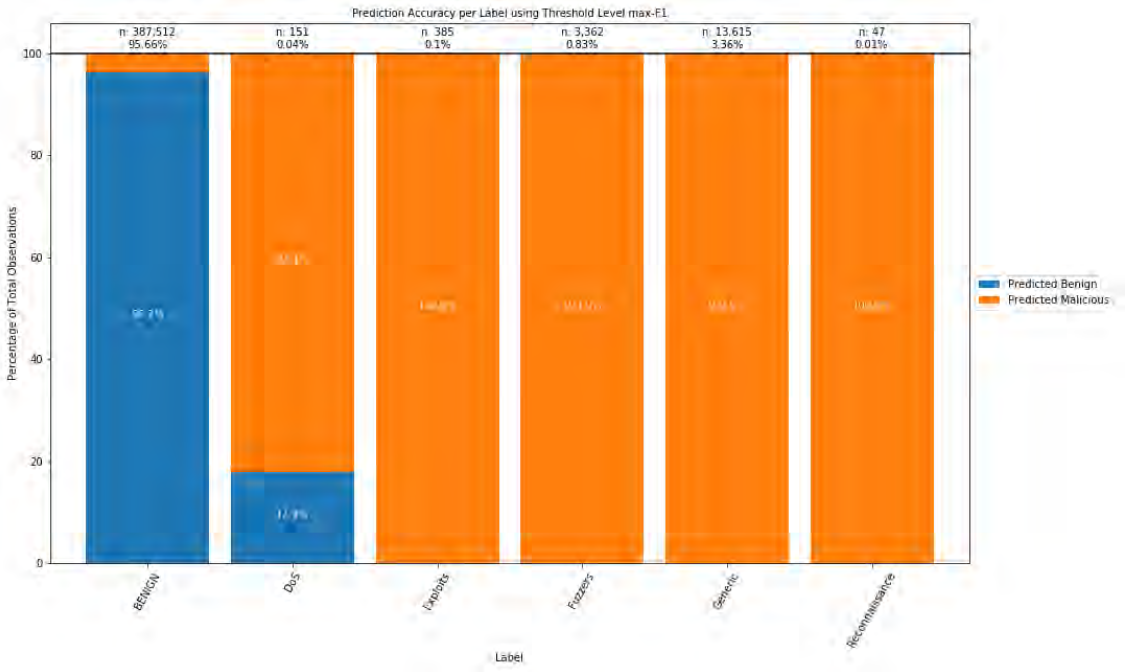


Figure 9.11: UNSW-NB15 DNS PCA_0.5 prediction per class

Optimal recall threshold

The last result which is given for the anomaly detection algorithms is the optimal recall threshold for the UNSW-NB15 dataset. Table 9.23 shows the threshold levels for which the recall is optimal. The UDP, DNS, FTP and SSH seem to require a low threshold level to attain this recall performance level. Eventhough for the SSH PCA models the recall is already 1 at 0.421% of all instances, the highest attained F_1 score seen in Table 9.22 is only 0.17167. The low F_1 score does not tell that 99.5% of the lowest anomaly scores were actually normal instances, while this stastic does tell so.

Table 9.23: UNSW-NB15 optimal recall thresholds

Model	**TCP**	**UDP**	**DNS**	**HTTP**	**FTP**	**SSH**
Anomaly Ratio	3.609%	2.31%	4.335%	5.0%	2.681%	0.04%
IForest_5	99.879%	13.311%	10.149%	99.337%	6.157%	6.046%
IForest_10	**88.77%**	**8.865%**	9.545%	**41.826%**	4.583%	6.025%
PCA_0.1	94.796%	69.833%	**8.029%**	100.0%	3.982%	**0.421%**
PCA_0.5	100.0%	100.0%	8.032%	53.574%	**3.77%**	**0.421%**
PCA_0.9	100.0%	100.0%	8.034%	53.583%	3.847%	**0.421%**

9.3 Graph Level

In the last part of this research, we take a look at the graph-level performance. First the experimental setup is discussed, after which the results of the models are discussed. In this section, only the results for the CIC-IDS-2017 and the UNSW-NB15 are discussed. The results for the ISCX-IDS-2012 can be found in Appendix C.

9.3.1 Experimental Setup

Intrusion data is high dimensional. The methods considered in this research only use one weight value in the edges. Therefore, the connection data is transformed to an anomaly score, which describes the likeliness of the behaviour seen in the communication between the hosts. In the previous section, connection features where transformed to anomaly scores in an unsupervised learning manner. These anomaly scores are now used as weights for the links between the edges.

The dataset now consists of two communicating hosts with data as start-time (and possibly an ending-time, e.g. TCP connections) and an anomaly score indicating the likeliness of the behaviour. To construct graph G_t, only the connections at a certain t are considered. Let t_0 be the first connection of the day and t_T the last time moment. It holds that $t \in \{t_0, t_0+\delta, t_0+2\cdot\delta, ..., t_T\}$ When connections in a certain protocol dataset have an starting time and an ending time, the data for connection t is extracted as follows.

$$X_t = \{x_i | t \leq x_{i,ts} \cap x_{i,ts\_} < t + \delta\}$$

There are protocols without the ending time feature. For these protocols the connections are extracted in the following way.

$$X_t = \{x_i | t \leq x_{i,ts} \cap x_{i,ts} < t + \delta\}$$

Of course it is possible to have multiple connections between two hosts at the same time. Therefore, we compute the maximum value for each duplicate connection. This results then in G_t. Each graph has either a normal or malicious label. This label is constructed by looking at the connections in X_t and look if one of the connections has a malicious label. If this is the case, then the graph is considered malicious, while is all connections are normal, the graph is considered normal. In this research, the time unit δ is taken 2 min after considering several options such as 5 seconds, 1 minute, 2 minutes, 10 minutes and 60 minutes.

This resulted in new datasets with the intrusion ratio's given in Table 9.24. This table shows how many network changes are labelled as malicious in relative perspective with the total number of changes of the network. The UNSW-NB15 has relatively many changes in the network with many malicious changes.

Table 9.24: Graph-based datasets intrusion ratio statistics

Dataset	**TCP**	**UDP**	**DNS**	**HTTP**	**FTP**	**SSH**	**SSL**
CIC-IDS-2017	0.32734	0.02167	0.018	0.09393	0.00333	0.03394	0.00041
ISCX-IDS-2012	0.08929	0.0003	0.0001	0.00474	0.0003	0.00491	0.0
UNSW-NB15	0.54257	0.54632	0.35453	0.54023	0.40636	0.01366	0.17486

In the following sections, the results for the applied distance metrics are given for the CIC-IDS-2017 dataset and the UNSW-NB15 dataset.

9.3.2 CIC-IDS-2017

To get results for the CIC-IDS-2017, the network change with the highest distances are labelled as malicious and the others are labelled normal. The intrusion ratio is used as a threshold to differentiate between the normal network change and the possibly malicious. Table 9.25 shows the results for the unweighted metrics. The TCP dataset is the dataset in which the NED distance is best able to capture the malicious network changes in the highest distance metrics.

Table 9.25: CIC-IDS-2017 unweighted edges methods F_1 score

Metric	TCP	UDP	DNS	HTTP	FTP	SSH	SSL
MCSED	0.2757	0.0	**0.04167**	0.04762	0.0	0.0	0.0
MCSVD	0.30607	0.0	**0.04167**	0.03968	0.0	0.0	0.0
NED	**0.37617**	0.0	**0.04167**	**0.15873**	**0.125**	**0.10638**	0.0
VEO	0.31075	**0.03333**	**0.04167**	0.01587	0.0	0.0	0.0

Table 9.26 shows the results for the weighted edges distance metrics. It can be observed that the weighted versions outperform the unweighted metrics in all protocols. Interesting is that in the SSH protocol the F_1 score is even 0.76596. On the TCP level, only a slight improvement is achieved with the Umeyama distance with IForest_5 anomaly scores as edge weights.

Table 9.26: CIC-IDS-2017 edge weighted methods F_1 score

Metric	Weight	TCP	UDP	DNS	HTTP	FTP	SSH	SSL
ED	IForest_10	0.29206	**0.03333**	**0.04167**	0.0	0.0	0.0	0.0
ED	IForest_5	0.29206	**0.03333**	**0.04167**	0.0	0.0	0.0	0.0
ED	PCA_0.1	0.29206	**0.03333**	**0.04167**	0.0	0.0	0.0	0.0
ED	PCA_0.5	0.29206	**0.03333**	**0.04167**	0.0	0.0	0.0	0.0
ED	PCA_0.9	0.29206	**0.03333**	**0.04167**	0.0	0.0	0.0	0.0
MCSWD	IForest_10	0.34112	0.0	0.0	0.07143	0.0	0.46809	0.0
MCSWD	IForest_5	0.36449	0.0	0.0	0.09524	0.0	0.48936	0.0
MCSWD	PCA_0.1	0.35748	0.0	0.0	0.13492	0.0	0.19149	0.0
MCSWD	PCA_0.5	0.36916	0.0	0.0	0.13492	0.0	0.34043	0.0
MCSWD	PCA_0.9	0.36916	0.0	0.0	0.13492	0.0	0.34043	0.0
UD	IForest_10	0.38084	0.0	**0.04167**	0.15873	0.0	0.57447	0.0
UD	IForest_5	**0.38318**	**0.03333**	**0.04167**	0.15079	**0.125**	0.51064	0.0
UD	PCA_0.1	0.37383	0.0	**0.04167**	**0.16667**	0.0	**0.76596**	0.0
UD	PCA_0.5	0.37617	0.0	**0.04167**	**0.16667**	0.0	0.61702	0.0
UD	PCA_0.9	0.3785	0.0	**0.04167**	**0.16667**	**0.125**	0.57447	0.0
WD	IForest_10	0.29907	0.0	**0.04167**	0.0	0.0	0.0	0.0
WD	IForest_5	0.29907	0.0	**0.04167**	0.0	0.0	0.0	0.0
WD	PCA_0.1	0.30374	0.0	**0.04167**	0.0	0.0	0.0	0.0
WD	PCA_0.5	0.30374	0.0	**0.04167**	0.0	0.0	0.0	0.0
WD	PCA_0.9	0.30374	0.0	**0.04167**	0.0	0.0	0.0	0.0

9.3.3 UNSW-NB15

This section gives the results of the graph-based approach on the UNSW-NB15 dataset. Table 9.27 shows the results for the unweighted distance metrics. It is clearly that the scores of these metrics are already high for all protocols except the SSH protocol. The MCSVD seems to be the best metric for this dataset.

Table 9.27: UNSW-NB15 unweighted F_1 score

Metric	**TCP**	**UDP**	**DNS**	**HTTP**	**FTP**	**SSH**	**SSL**
MCSED	0.96139	0.96287	0.53053	0.95745	0.61231	0.0	0.53125
MCSVD	**0.96264**	**0.96782**	**0.92176**	**0.9587**	**0.68719**	0.0	0.53125
NED	0.85554	0.95792	0.44275	0.9224	0.64559	0.0	**0.58594**
VEO	0.82441	0.93564	0.42557	0.69086	0.52246	0.0	0.49219

At last, the results for the weighted distance metrics are found in Table 9.28. Clearly, the Umeyama distance metrics seem to outperform other distance metrics in all protocol datasets. Still the unweighted MCSVD seem to outperform the weighted metrics for most protocols, such as TCP, DNS, HTTP and FTP.

Table 9.28: UNSW-NB15 edge weighted F_1 scores

Metric	**Weight**	**TCP**	**UDP**	**DNS**	**HTTP**	**FTP**	**SSH**	**SSL**
WD	PCA_0.1	0.74346	0.87748	0.35305	0.73467	0.41764	0.0	0.53125
WD	PCA_0.5	0.78082	0.87995	0.35687	0.72966	0.42097	0.0	0.53125
WD	PCA_0.9	0.78829	0.88366	0.35496	0.72716	0.43428	0.0	0.53125
WD	IForest_5	0.78829	0.86386	0.35115	0.74844	0.41098	0.0	0.53125
WD	IForest_10	0.78829	0.86757	0.34351	0.7184	0.41431	0.0	0.53125
ED	PCA_0.1	0.6538	0.23391	0.09924	0.53442	0.12313	0.0	0.19531
ED	PCA_0.5	0.63885	0.23391	0.27099	0.53442	0.1198	0.0	0.19531
ED	PCA_0.9	0.63885	0.23391	0.27481	0.53442	0.1198	0.0	0.19531
ED	IForest_5	0.64259	0.23391	0.19084	0.53942	0.12146	0.0	0.19531
ED	IForest_10	0.64882	0.23391	0.2729	0.53692	0.12146	0.0	0.19531
MCSWD	PCA_0.1	0.45205	0.4802	0.31679	0.57947	0.38769	0.0	0.20312
MCSWD	PCA_0.5	0.44956	0.47525	0.28244	0.55444	0.37438	0.0	0.20312
MCSWD	PCA_0.9	0.44956	0.47772	0.28626	0.55319	0.38935	0.0	0.20312
MCSWD	IForest_5	0.47447	0.54084	0.29389	0.54318	0.38103	0.0	0.20312
MCSWD	IForest_10	0.48319	0.52847	0.2958	0.49061	0.37438	0.0	0.20312
UD	PCA_0.1	0.73724	0.9604	**0.53053**	**0.95369**	**0.61398**	0.0	0.60156
UD	PCA_0.5	0.79452	**0.96782**	0.51718	0.94994	0.56739	0.0	**0.63281**
UD	PCA_0.9	0.80199	0.96535	0.45038	0.94618	0.54576	0.0	0.60938
UD	IForest_5	**0.82565**	0.95916	0.41031	0.94243	0.53245	0.0	0.60938
UD	IForest_10	0.82192	0.96535	0.43702	0.94493	0.53744	0.0	0.60938

Chapter 10

Discussion and Conclusion

In this first part of this research, the assumption that malicious traffic is indeed different from normal traffic is checked by applying supervised learning techniques. Multiple data sources describing not merely the transport layer usage but also application layer protocols helped the machine learning techniques distinguishing attacks from normal traffic. One example which shows this are the web attacks in the CIC-IDS-2017 dataset. The F_1 performance for these attacks are boosted by using the HTTP dataset over the TCP dataset.

One main contribution in this research are the new and improved labeling schemes for the ISCX-IDS-2012 and the CIC-IDS-2017 dataset. These labelling schemes can be used instead of the published labelling schemes for more accurately labelling the connections of the dataset. Improving labelling schemes helps the research community as the data is better labelled.

In the second part of this research, the unsupervised learning techniques are applied on the same datasets to see whether assigning anomaly scores to connections can help detecting malicious traffic in an anomaly detection manner. Unfortunately, the unsupervised learning techniques which are applied did not result in F_1 scores outperforming the benchmark in many cases. This motivates the idea of using the graph-structure of the data to also take host correlations into account.

In the last part of this research, we have applied several graph distance metrics to look at the change of the network over time to detect the moments in which attacks occur. Unfortunately, for the CIC-IDS-2017 and the ISCX-IDS-2012 dataset the results were poorly in many cases. However, the performance of the models on the USNW-NB15 dataset has outperformed the baselines with an F_1 score of 0.96785 for the Umeyama distance with PCA_0.5 as edge weights for the UDP protocol and a F_1 score of 0.95369 for the same distance metric with PCA_0.1 as edge weights for the HTTP dataset. However, the unweighted MCSVD metric outperformed the Umeyama distance metrics for the TCP, DNS, HTTP and FTP datasets. The structure of the network has a great influence on the distances measuring the network change.

Future work

As the title of this research already indicates, this research is not a closed book. There are many methods in graph-based intrusion detection and only a small part of research within this field is studied in this research. In this research, the change of the network is labelled as malicious, but there are also methods in which nodes are labelled as malicious, edges or sub-graphs. Applying other graph-based methods on the latest intrusion detection dataset can determine which methods are promising and which are not.

References

Aggarwal, C. C. (2015a). *Data Mining*. Cham: Springer International Publishing. Retrieved from http://link.springer.com/10.1007/978-3-319-14142-8 doi: 10.1007/978-3-319-14142-8

Aggarwal, C. C. (2015b). Outlier analysis. In *Data mining* (pp. 237–263).

Akoglu, L., McGlohon, M., & Faloutsos, C. (2010). oddball: Spotting Anomalies in Weighted Graphs. In (pp. 410–421). Retrieved from http://www.cs.cmu.edu/{~}lakoglu/pubs/OddBall{_}cameraready.pdf{%}5Cnpapers://63650f89-9a27-4a9d-91f7-fca3c5048b3e/Paper/p1946http://link.springer.com/10.1007/978-3-642-13672-6{_}40 doi: 10.1007/978-3-642-13672-6_40

Akoglu, L., Tong, H., & Koutra, D. (2015, may). Graph based anomaly detection and description: a survey. *Data Mining and Knowledge Discovery*, *29*(3), 626–688. Retrieved from http://link.springer.com/10.1007/s10618-014-0365-y doi: 10.1007/s10618-014-0365-y

Axelsson, S. (2000). *Intrusion Detection Systems : A Survey and Taxonomy* (Tech. Rep.). Technical report.

Berlingerio, M., Koutra, D., Eliassi-Rad, T., & Faloutsos, C. (2012). *NetSimile: A Scalable Approach to Size-Independent Network Similarity*. Retrieved from http://arxiv.org/abs/1209.2684

Berners-Lee, T., Fielding, R. T., & Masinter, L. (2005, January). *Uniform resource identifier (uri): Generic syntax* (STD No. 66). RFC Editor. Internet Requests for Comments. Retrieved from http://www.rfc-editor.org/rfc/rfc3986.txt (http://www.rfc-editor.org/rfc/rfc3986.txt)

Bhuyan, M. H., Bhattacharyya, D. K., & Kalita, J. K. (2015). Towards generating real-life datasets for network intrusion detection. *I. J. Network Security*, *17*, 683-701.

Bonaventure, O., et al. (2011). *Computer networking: Principles, protocols and practice*. Citeseer.

Breiman, L. (2001). Random Forests. *Machine Learning*, *45*(6), 5—-32. doi: 10.1023/A:1010933404324

Bullar, C., & Gerth, J. (2014). *Pcr - a new flow metric*. Retrieved from https://qosient.com/argus/presentations/Argus.FloCon.2014.PCR.Presentation.pdf

cbronline. (n.d.). https://www.cbronline.com/digital-transformation/lessons-learned-notpetya/. (Online; accessed 11 March 2020)

Chalapathy, R., & Chawla, S. (2019). *Deep learning for anomaly detection: A survey*.

Chandola, V., Banerjee, A., & Kumar, V. (2007). Outlier detection: A survey. *ACM Computing Surveys*, *14*, 15.

Chandola, V., Banerjee, A., & Kumar, V. (2009, jul). Anomaly detection. *ACM Computing Surveys*, *41*(3), 1–58. Retrieved from http://portal.acm.org/citation.cfm?doid=1541880.1541882 doi: 10.1145/1541880.1541882

Charte, F., Rivera Rivas, A., Del Jesus, M. J., & Herrera, F. (2016, 04). On the impact of dataset complexity and sampling strategy in multilabel classifiers performance. In (p. 500-511). doi: 10.1007/978-3-319-32034-2_42

Cheung, D. (2020). *Common dns request types*. https://support.opendns.com/hc/en-us/articles/227986607-Common-DNS-Request-Types. (Online; accessed 13 March 2020)

Coull, S., Wright, C., Monrose, F., Collins, M., & Reiter, M. (2007, 01). Playing devil's advocate: Inferring sensitive information from anonymized network traces..

David, G. (2017, Apr). *Firewalls: Still your first line of defense.* Retrieved from https://securingtomorrow.mcafee.com/consumer/consumer-threat-notices/firewalls-still-first-line-defense/

Day, J. D., & Zimmermann, H. (1983, Dec). The osi reference model. *Proceedings of the IEEE*, *71*(12), 1334-1340. doi: 10.1109/PROC.1983.12775

Debar, H., Dacier, M., & Wespi, A. (1999, apr). Towards a taxonomy of intrusion-detection systems. *Computer Networks*, *31*(8), 805–822. doi: 10.1016/S1389-1286(98)00017-6

Deering, S. E., & Hinden, R. M. (1998, December). *Internet protocol, version 6 (ipv6) specification* (RFC No. 2460). RFC Editor. Internet Requests for Comments. Retrieved from http://www.rfc-editor.org/rfc/rfc2460.txt (http://www.rfc-editor.org/rfc/rfc2460.txt)

Denning, D. (1987, feb). An intrusion-detection model. *IEEE Transactions on Software Engineering*, *SE-13*(2), 222–232. doi: 10.1109/TSE.1987.232894

Dickinson, P., & Kraetzl, M. (2003, 02). Novel approaches in modelling dynamics of networked surveillance environment. In (Vol. 1, p. 302 - 309). doi: 10.1109/ICIF.2003.177461

Dusseault, L., & Snell, J. (2010, March). *Patch method for http* (RFC No. 5789). RFC Editor. Internet Requests for Comments. Retrieved from http://www.rfc-editor.org/rfc/rfc5789.txt (http://www.rfc-editor.org/rfc/rfc5789.txt)

Fielding, R., & Reschke, J. (2014, June). *Hypertext transfer protocol (http/1.1): Semantics and content* (RFC No. 7231). RFC Editor. Internet Requests for Comments. Retrieved from http://www.rfc-editor.org/rfc/rfc7231.txt (http://www.rfc-editor.org/rfc/rfc7231.txt)

Floyd, S., & Paxson, V. (2001). Difficulties in simulating the Internet. *IEEE/ACM Transactions on Networking*, *9*(4), 392–403. Retrieved from http://ieeexplore.ieee.org/document/944338/ doi: 10.1109/90.944338

Freund, Y., & Schapire, R. E. (1997, aug). A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting. *Journal of Computer and System Sciences*, *55*(1), 119–139. Retrieved from http://link.springer.com/10.1007/3-540-59119-2{_}166https://linkinghub.elsevier.com/retrieve/pii/S002200009791504X doi: 10.1006/jcss.1997.1504

Gaston, M., Kraetzl, M., & Wallis, W. (2006, 01). Using graph diameter for change detection in dynamic networks. *The Australasian Journal of Combinatorics [electronic only]*, *35*.

Greenberg, A. (2018). *The untold story of notpetya, the most devastating cyberattack in history.* Retrieved from https://www.wired.com/story/notpetya-cyberattack-ukraine-russia-code-crashed-the-world/

Grigorik, I. (2013). *High performance browser networking: What every web developer should know about networking and web performance.* O'Reilly Media. Retrieved from https://www.xarg.org/ref/a/1449344763/

Gupta, M., Gao, J., Aggarwal, C., & Han, J. (2014). Outlier detection for temporal data. *Synthesis Lectures on Data Mining and Knowledge Discovery*, *5*(1), 1–129.

Habeeb, R. A. A., Nasaruddin, F., Gani, A., Hashem, I. A. T., Ahmed, E., & Imran, M. (2019). Real-time big data processing for anomaly detection: A survey. *International Journal of Information Management*, *45*, 289–307.

Hock-Chuan, C. (2009). *Http (hypertext transfer protocol).* https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html. (Online; accessed 12 March 2020)

Howard, M., & Leblanc, D. E. (2002). *Writing secure code* (2nd ed.). Redmond, WA, USA: Microsoft Press.

ITU. (2017). *Ict facts and figures 2017* (Tech. Rep.). Retrieved from https://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2017.pdf

Klein, J., Bhulai, S., Hoogendoorn, M., Van Der Mei, R., & Hinfelaar, R. (2018, dec). Detecting Network Intrusion beyond 1999: Applying Machine Learning Techniques to a Partially Labeled Cybersecurity Dataset. In *2018 ieee/wic/acm international conference on web intelligence (wi)* (pp. 784–787). IEEE. Retrieved from https://ieeexplore.ieee.org/document/8609692/ doi: 10.1109/WI.2018.00017

Koutra, D., Vogelstein, J. T., & Faloutsos, C. (2013, may). DeltaCon: A Principled Massive-Graph Similarity Function. In *Proceedings of the 2013 siam international conference on data mining* (Vol. 10, pp. 162–170). Philadelphia, PA: Society for Industrial and Applied Mathematics. Retrieved from http://dl.acm.org/citation.cfm?doid=2888412.2824443https://epubs.siam.org/doi/10.1137/1.9781611972832.18 doi: 10.1137/1.9781611972832.18

Kurose, J. F., & Ross, K. W. (2012). *Computer networking: A top-down approach (6th edition)* (6th ed.). Pearson.

Lashkari, A. (2019). *Cicflowmeter*. https://github.com/ahlashkari/CICFlowMeter. GitHub.

Le, D. Q., Jeong, T., Roman, H. E., & Hong, J. W.-K. (2011). Traffic dispersion graph based anomaly detection. In *Proceedings of the second symposium on information and communication technology - soict '11* (p. 36). New York, New York, USA: ACM Press. Retrieved from http://dl.acm.org/citation.cfm?doid=2069216.2069227 doi: 10.1145/2069216.2069227

Leuth, K. L. (2018). *State of the iot 2018: Number of iot devices now at 7b – market accelerating*. Retrieved from https://iot-analytics.com/state-of-the-iot-update-q1-q2-2018-number-of-iot-devices-now-7b/

Liao, H.-J., Richard Lin, C.-H., Lin, Y.-C., & Tung, K.-Y. (2013, jan). Intrusion detection system: A comprehensive review. *Journal of Network and Computer Applications*, *36*(1), 16–24. Retrieved from http://dx.doi.org/10.1016/j.jnca.2012.09.004 doi: 10.1016/j.jnca.2012.09.004

Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008, dec). Isolation forest. In *2008 eighth ieee international conference on data mining* (pp. 413–422). IEEE. doi: 10.1109/ICDM.2008.17

Mockapetris, P. (1987, November). *Domain names - implementation and specification* (STD No. 13). RFC Editor. Internet Requests for Comments. Retrieved from http://www.rfc-editor.org/rfc/rfc1035.txt (http://www.rfc-editor.org/rfc/rfc1035.txt)

Moustafa, N., & Slay, J. (2015, nov). UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In *2015 military communications and information systems conference (milcis)* (pp. 1–6). IEEE. doi: 10.1109/MilCIS.2015.7348942

Mudzingwa, D., & Agrawal, R. (2012, mar). A study of methodologies used in intrusion detection and prevention systems (IDPS). In *2012 proceedings of ieee southeastcon* (pp. 1–6). IEEE. doi: 10.1109/SECon.2012.6197080

Murty, M. N., & Devi, V. S. (2011). *Pattern Recognition* (Vol. 24). London: Springer London. Retrieved from http://link.springer.com/10.1007/978-0-85729-495-1 doi: 10.1007/978-0-85729-495-1

NCTV. (2018). *Cybersecuritybeeld nederland* (Tech. Rep.). CSBN 2018.

Papadimitriou, P., Dasdan, A., & Garcia-Molina, H. (2010). Web graph similarity for anomaly detection. *Journal of Internet Services and Applications*, *1*(1), 19–30. doi: 10.1007/s13174-010-0003-x

Pincombe, B. (2005). Anomaly detection in time series of graphs using arma processes. *Asor Bulletin*, *24*(4), 2.

Postel, J. (1980, August). *User datagram protocol* (STD No. 6). RFC Editor. Internet Requests for Comments. Retrieved from http://www.rfc-editor.org/rfc/rfc768.txt (http://www.rfc-editor.org/rfc/rfc768.txt)

Postel, J. (1981a, September). *Internet protocol* (STD No. 5). RFC Editor. Internet Requests for Comments. Retrieved from http://www.rfc-editor.org/rfc/rfc791.txt

Postel, J. (1981b, September). *Transmission control protocol* (STD No. 7). RFC Editor. Internet Requests for Comments. Retrieved from http://www.rfc-editor.org/rfc/rfc793.txt (http://www.rfc-editor.org/rfc/rfc793.txt)

Postel, J., & Reynolds, J. (1985, October). *File transfer protocol* (STD No. 9). RFC Editor. Internet Requests for Comments. Retrieved from http://www.rfc-editor.org/rfc/rfc959.txt (http://www.rfc-editor.org/rfc/rfc959.txt)

Powers, D. M. (2011). Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation.

Ren, Q., Li, M., & Han, S. (2019, 02). Tectonic discrimination of olivine in basalt using data mining techniques based on major elements: a comparative study from multiple perspectives. *Big Earth Data*, 1-18. doi: 10.1080/20964471.2019.1572452

Ring, M., Wunderlich, S., Scheuring, D., Landes, D., & Hotho, A. (2019, sep). A survey of network-based intrusion detection data sets. *Computers & Security*, *86*, 147–167. Retrieved from http://dx.doi.org/10.1016/j.cose.2019.06.005 doi: 10.1016/j.cose.2019.06.005

Scarfone, K., & Mell, P. (2007). *Guide to Intrusion Detection and Prevention Systems (IDPS)* (Tech. Rep.). Gaithersburg, MD, United States.

Sharafaldin, I., Habibi Lashkari, A., & Ghorbani, A. (2018, 01). Toward generating a new intrusion detection dataset and intrusion traffic characterization. In (p. 108-116). doi: 10.5220/0006639801080116

Shiravi, A., Shiravi, H., Tavallaee, M., & Ghorbani, A. A. (2012, may). Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security*, *31*(3), 357–374. Retrieved from http://dx.doi.org/10.1016/j.cose.2011.12.012 doi: 10.1016/j.cose.2011.12.012

Showbridge, P., Kraetzl, M., & Ray, D. (1999). Detection of abnormal change in dynamic networks. In *1999 information, decision and control. data and information fusion symposium, signal processing and communications symposium and decision and control symposium. proceedings (cat. no.99ex251)* (pp. 557–562). IEEE. Retrieved from http://ieeexplore.ieee.org/document/754216/ doi: 10.1109/IDC.1999.754216

Shyu, M.-L., Chen, S.-C., Sarinnapakorn, K., & Chang, L. (2003). *A novel anomaly detection scheme based on principal component classifier* (Tech. Rep.). MIAMI UNIV CORAL GABLES FL DEPT OF ELECTRICAL AND COMPUTER ENGINEERING.

Sisco, A. (2018). *Your first line of defense: Firewalls.* Retrieved from https://www.coretech.us/blog/your-first-line-of-defense-firewalls

Sommer, R., & Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. *Proceedings - IEEE Symposium on Security and Privacy*, 305–316. doi: 10.1109/SP.2010.25

Tanenbaum, A. S., & Wetherall, D. J. (2011). *Computer networks* (5th ed.). Upper Saddle River, NJ, USA: Prentice Hall Press.

Thomas, C., Sharma, V., & Balakrishnan, N. (2008). Usefulness of darpa dataset for intrusion detection system evaluation. In *Data mining, intrusion detection, information assurance, and data networks security.*

Wang, Z., Zhang, J., & Verma, N. (2015, 12). Realizing low-energy classification systems by implementing matrix multiplication directly within an adc. *IEEE Transactions on Biomedical Circuits and Systems*, *9*, 1-1. doi: 10.1109/TBCAS.2015.2500101

Will Koehrsen. (2017). *Random Forest Simple Explanation.* https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d. (Online; accessed 19 March 2020)

Xinyou Zhang, Chengzhong Li, & Wenbin Zheng. (2004). Intrusion prevention system design. In *The fourth international conference oncomputer and information technology, 2004. cit '04.* (pp. 386–390). IEEE. Retrieved from http://ieeexplore.ieee.org/document/1357226/ doi: 10.1109/CIT.2004.1357226

Yon Tang, & Shigang Chen. (2005). Defending against internet worms: a signature-based approach. In *Proceedings ieee 24th annual joint conference of the ieee computer and communications societies.* (Vol. 2, pp. 1384–1394). IEEE. doi: 10.1109/INFCOM.2005.1498363

Appendices

Appendix A

ISCX-IDS-2012 Meta Data

Table A.1: ISCX-IDS-2012 Network Addresses

IPv4 Private	MAC Addresses	ARP Addresses
192.168.1.101	00:11:25:bb:1f:cf	Ibm_bb:1f:cf
192.168.1.102	00:11:25:bb:ce:a1	Ibm_bb:ce:a1
192.168.1.103	00:11:25:bb:ce:df	Ibm_bb:ce:df
192.168.1.104	00:11:25:b9:ac:22	Ibm_b9:ac:22
192.168.1.105	00:11:25:5f:9b:4f	Ibm_5f:9b:4f
192.168.2.106	00:11:25:b9:a7:ed	Ibm_b9:a7:ed
192.168.2.107	00:11:25:bb:d2:6f	Ibm_bb:d2:6f
192.168.2.108	00:02:55:bf:75:a9	Ibm_bf:75:a9
192.168.2.109	00:09:6b:03:e8:a4	Ibm_03:e8:a4
192.168.2.110	00:09:6b:e9:a9:00	Ibm_e9:a9:00
192.168.2.111	00:09:6b:8b:63:8e	Ibm_8b:63:8e
192.168.2.112	00:09:6b:e9:cf:6b	Ibm_e9:cf:6b
192.168.2.113	00:09:6b:e9:d0:88	Ibm_e9:d0:88
192.168.3.114	00:0d:60:96:ac:5e	Ibm_96:ac:5e
192.168.3.115	00:0d:60:96:ac:f7	Ibm_96:ac:f7
192.168.3.116	00:0d:60:96:ad:25	Ibm_96:ad:25
192.168.3.117	00:09:6b:d8:7f:9a	Ibm_d8:7f:9a
192.168.4.118	00:0d:60:96:ac:f6	Ibm_96:ac:f6
192.168.4.119	00:02:55:7f:70:de	Ibm_7f:70:de
192.168.4.120	00:11:25:bb:cf:0f	Ibm_bb:cf:0f
192.168.4.121	00:0d:60:96:ae:5c	Ibm_96:ae:5c
192.168.5.122	00:22:19:20:7b:d3	Dell_20:7b:d3
192.168.5.123	00:09:6b:d8:f2:c8	Ibm_d8:f2:c8
192.168.5.124	00:01:02:72:ab:55	3Com_72:ab:55

Table A.2: ISCX-IDS-2012 Improved Labeling Scheme

Day	Start	End	Orig IP	Resp IP	Orig Port	Resp Port	Label
13-6	15:32	15:32	131.202.243.90	192.168.5.122	5096	25	Infiltration
13-6	16:11	21:00	192.168.1.105	131.202.243.90	54073	5555	Infiltration
13-6	16:50	21:02	192.168.2.112	131.202.243.90	3542	5555	Infiltration
13-6	16:37	16:39	192.168.1.105	192.168.1.			Portscan
13-6	16:42	16:44	192.168.1.105	192.168.2.			Portscan
13-6	16:50	18:31	192.168.1.105	192.168.2.112		445	Exploit
13-6	16:58	16:59	192.168.2.112	192.168.5.			Portscan
13-6	17:03	18:31	192.168.1.105	192.168.2.112		4444	Exploit
(13-18)-6	20:59	00:08	192.168.2.112	131.202.243.84		5555	Backdoor
13-6	22:43	22:50	142.167.88.44	192.168.5.122			SQL Password
14-6	17:19	17:20	192.168.1.112	192.168.3.	52707		Portscan
14-6	17:19	17:20	192.168.1.112	192.168.3.	52708	443	Portscan
14-6	17:19	17:20	192.168.1.112	192.168.3.		143	Portscan
14-6	17:21	17:22	192.168.2.112	192.168.2.113		445	Vulnerable SMB
14-6	17:21	17:37	192.168.2.112	192.168.2.113		4444	Exploit
14-6	17:28	17:36	192.168.2.113	192.168.5.122		80	DDoS slowloris
14-6	17:39	17:50	192.168.2.112	192.168.3.115		445	Vulnerable SMB
14-6	17:39	17:50	192.168.2.112	192.168.3.115		4444	Exploit
14-6	17:43	17:49	192.168.3.115	192.168.5.122		80	DDoS slowloris
14-6	17:52	18:00	192.168.2.112	192.168.3.117		445	Vulnerable SMB
14-6	17:52	18:00	192.168.2.112	192.168.3.117		4444	Exploit
14-6	18:03	18:14	192.168.2.112	192.168.2.106		445	Vulnerable SMB
14-6	18:03	18:14	192.168.2.112	192.168.2.106		4444	Exploit
14-6	18:07	18:14	192.168.2.106	192.168.5.122		80	DDoS slowloris
14-6	18:15	18:26	192.168.2.112	192.168.1.101		445	Vulnerable SMB
14-6	18:15	18:26	192.168.2.112	192.168.1.101		4444	Exploit
14-6	18:19	18:26	192.168.1.101	192.168.5.122		80	DDoS slowloris
15-6	15:27	17:06	192.168.1.105	192.168.2.112		6667	Backdoor
15-6	15:33	17:06	192.168.2.109	192.168.2.112		6667	Backdoor
15-6	15:37	17:06	192.168.4.118	192.168.2.112		6667	Backdoor
15-6	15:41	17:06	192.168.2.113	192.168.2.112		6667	Backdoor
15-6	15:51	17:06	192.168.1.103	192.168.2.112		6667	Backdoor
15-6	15:53	17:06	192.168.2.110	192.168.2.112		6667	Backdoor
15-6	15:59	17:06	192.168.4.120	192.168.2.112		6667	Backdoor
15-6	16:04	17:07	192.168.1.103	192.168.5.122		80	DDoS Botnet
15-6	16:04	17:07	192.168.1.105	192.168.2.122		80	DDoS Botnet
15-6	16:04	17:07	192.168.2.109	192.168.5.122		80	DDoS Botnet
15-6	16:04	17:07	192.168.2.110	192.168.5.122		80	DDoS Botnet
15-6	16:04	17:07	192.168.2.113	192.168.5.122		80	DDoS Botnet
15-6	16:04	17:07	192.168.4.118	192.168.5.122		80	DDoS Botnet
15-6	16:04	17:07	192.168.4.120	192.168.5.122		80	DDoS Botnet
17-6	14:25	15:01	131.202.243.90	192.168.5.122		22	SSH Brute Force
17-6	15:02	17:12	131.202.243.90	192.168.5.122	9930	22	Exfiltration

Appendix B

CIC-IDS-2017 Meta Data

Table B.1: CIC-IDS-2017 Network Addresses

IPv4 Private	**IPv4 Public**	**IPv6**	**MAC Addresses**
172.16.0.1	205.174.165.80		
192.168.10.1			
192.168.10.3		fe80::3109:ba1d:7470:7dd	
192.168.10.5			
192.168.10.8		fe80::6d07:ea13:6f73:ed41	
192.168.10.9			
192.168.10.12		fe80::8cc1:7756:5ffc:823f	
192.168.10.14		fe80::3c60:fc97:3f94:ac8f	
192.168.10.15		fe80::f470:cd16:3cf6:bd7b	
192.168.10.16		fe80::cd08:22a7:a40f:3d73	
192.168.10.17		fe80::223:aeff:fe9b:9567	00:23:ae:9b:95:67
192.168.10.19		fe80::223:aeff:fe9b:adb3	00:23:ae:9b:ad:b3
192.168.10.25			00:25:00:a8:c4:60
192.168.10.50	205.174.165.68	fe80::3e30:c3c4:d72f:ae4c	
192.168.10.51	205.174.165.66	fe80::baac:6fff:fe36:ba8	b8:ac:6f:36:0b:a8

Table B.2: CIC-IDS-2017 Improved Labeling Scheme

Day	Start	End	Orig IP	Resp IP	Orig Port	Resp Port	Label
4-7	09:16	10:21	172.16.0.1	192.168.10.50		21	Patator - FTP
4-7	14:08	15:12	172.16.0.1	192.168.10.50		22	Patator - SSH
5-7	09:47	10:12	172.16.0.1	192.168.10.50		80	DoS Slowloris
5-7	10:14	10:38	172.16.0.1	192.168.10.50		80	DoS SlowHTTPTest
5-7	10:42	11:08	172.16.0.1	192.168.10.50		80	DoS Hulk
5-7	11:09	11:21	172.16.0.1	192.168.10.50		80	DoS Goldeneye
5-7	15:11	15:33	172.16.0.1	192.168.10.51		444	Heartbleed
6-7	09:14	10:01	172.16.0.1	192.168.10.50		80	Web Attack - Brute Force
6-7	10:14	10:36	172.16.0.1	192.168.10.50		80	Web Attack - XSS
6-7	10:39	10:43	172.16.0.1	192.168.10.50		80	Web Attack - Sql Injection
6-7	14:18	14:43	205.174.165.73	192.168.10.8	444		Infiltration - Dropbox download
6-7	14:18	14:43	192.168.10.8	192.168.10.5			Portscan
6-7	14:52	15:05	205.174.165.73	192.168.10.25	444		Infiltration - Cool disk - MAC
6-7	15:03	15:46	205.174.165.73	192.168.10.8	444		Infiltration - Dropbox download
6-7	15:03	15:46	192.168.10.8	192.168.10.			Portscan
7-7	10:04	17:03	205.174.165.73	192.168.10.5	8080		Botnet Ares
7-7	10:04	17:03	205.174.165.73	192.168.10.8	8080		Botnet Ares
7-7	10:04	17:03	205.174.165.73	192.168.10.9	8080		Botnet Ares
7-7	10:04	17:03	205.174.165.73	192.168.10.14	8080		Botnet Ares
7-7	10:04	17:03	205.174.165.73	192.168.10.15	8080		Botnet Ares
7-7	13:01	15:24	172.16.0.1	192.168.10.50			Port Scan
7-7	15:55	16:17	172.16.0.1	192.168.10.50		80	DDoS LOIC

Appendix C

ISCX-IDS-2012 Results Graph data

Table C.1: ISCX-IDS-2012

Metric	TCP	UDP	DNS	HTTP	FTP	SSH
MCSED	0.17889	0.0	0.0	0.0	0.0	0.0
MCSVD	0.18222	0.0	0.0	0.0	0.0	0.0
NED	0.04	0.0	0.0	0.0	0.0	0.0
VEO	0.08	0.0	0.0	0.0	0.0	0.0

Table C.2: ISCX-IDS-2012 table

Metric	Weight	TCP	UDP	DNS	HTTP	FTP	SSH
WD	PCA_0.1	0.10556	0.0	0.0	0.0	0.0	0.0
WD	PCA_0.5	0.10667	0.0	0.0	0.0	0.0	0.0
WD	PCA_0.9	0.10333	0.0	0.0	0.0	0.0	0.0
WD	IForest_5	0.10222	0.0	0.0	0.0	0.0	0.0
WD	IForest_10	0.10444	0.0	0.0	0.0	0.0	0.0
ED	PCA_0.1	0.11333	0.0	0.0	0.0	0.0	0.0
ED	PCA_0.5	0.11556	0.0	0.0	0.0	0.0	0.0
ED	PCA_0.9	0.11333	0.0	0.0	0.0	0.0	0.0
ED	IForest_5	0.11889	0.0	0.0	0.0	0.0	0.0
ED	IForest_10	0.11778	0.0	0.0	0.0	0.0	0.0
MCSWD	PCA_0.1	0.12778	0.0	0.0	0.0	0.0	0.04255
MCSWD	PCA_0.5	0.14	0.0	0.0	0.0	0.0	0.04255
MCSWD	PCA_0.9	0.13778	0.0	0.0	0.0	0.0	0.10638
MCSWD	IForest_5	0.11667	0.0	0.0	0.0	0.0	0.12766
MCSWD	IForest_10	0.12	0.0	0.0	0.0	0.0	0.12766
UD	PCA_0.1	0.04556	0.0	0.0	0.0	0.0	0.0
UD	PCA_0.5	0.04444	0.0	0.0	0.0	0.0	0.0
UD	PCA_0.9	0.04333	0.0	0.0	0.0	0.0	0.0
UD	IForest_5	0.03889	0.0	0.0	0.0	0.0	0.0
UD	IForest_10	0.04111	0.0	0.0	0.0	0.0	0.0