Exploring NAT Host Counting Using Network Traffic Flows

Sebastian Salomonsson

Faculty of Health, Science and Technology

Degree Project for Master of Science in Engineering

Supervisor: Johan Garcia

Examiner: Kerstin Andersson

30 Hp

2017-06-09

Abstract

Network Address Translation (NAT) is used to represent a private internal network which may consist of several hosts with a single outward IP-address. This is an important method used to distribute Internet access, as the IP-addresses provided by IPv4 are becoming scarce. NAT is therefore often used by home routers in order to provide Internet access for local devices. The private network behind a NAT becomes hidden from the public domain and only one IP-address may be visible from the private network. It is of Internet Service Providers (ISPs) and network operators interest to know how their services are used and with NAT it can be difficult to know how many hosts that are using their Internet service. This study will focus on analyzing data flows from real network traffic in order to identify indications of NAT behavior and then count the number of hosts in the NAT network. There exist several studies of how to determine the number of hosts behind a NAT. However, some of the methods rely on signatures in the header fields of communication protocols which can be unreliable and not possible to use when the traffic is encrypted. This study focuses primarily on the detection of large amounts of hosts which have a Windows Operating System (OS). An empirical study is made to identify the distribution and characteristics of Windows Update flows and the results are used in the NAT host counting methods. NAT characteristics were found in the analyzed datasets as well as some indication of a number of hosts. However to provide a higher accuracy on a number of hosts more research have to be made.

Keywords: NAT, NAT host counting, Data analysis, NAT detection using network traffic flows.

Acknowledgements

I would like to thank Procera Networks that gave me the opportunity to work with this highly interesting project. A special thank to Anders Waldenborg who was my supervisor at Procera and the one who provided me with some of the datasets. I would also like to thank Johan Garcia who was my supervisor at Karlstad university who helped me with the dissertation and gave me advices during the course of the project.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

The number of Internet-connected devices and users have increased dramatically during the last couple of years. Never before has the Internet usage and traffic been so large. It is estimated that around 3.6 billion users and 20.35 billion devices have access to the Internet today and it is still increasing [32], [33].

Internet Service Providers (ISP) provide Internet access to their customers, but with many users and devices connected it is in their interest to know how the connectivity is used and possibly redistributed. In order for all of these devices to get a connection to the Internet, each of them will have to be provided with their own unique IP-address. Unfortunately, the number of IP-addresses available from the existing Internet Protocol IPv4 are becoming scarce. There have been issues with the transition to the more modern IPv6 Protocol, which is why IPv4 is still used to carry the majority of network traffic but with the help of Network Address Translation (NAT).

NAT [29] can be used to represent a local network consisting of several devices with just a single or a pool of IP-addresses. This makes it possible for an ISP to distribute Internet access to its customers and all of their devices even if the IP-addresses provided by IPv4 is limited. NAT can also be used to provide security and anonymity by hiding the inner network topology, filter content etc. But even if NAT solves the problem with depleted IP-addresses other issues have been introduced. NAT can cause the ISP's to not know how their services are used or how large the inner network might be. Unauthorized NAT devices might be installed to illegally sell and redistribute an Internet connection. Therefore it is of an ISP's interest to know if a single IP-address might represent a large network in order to determinate if a customer is sharing their connection inappropriately with other hosts. Since NAT is able to represent a network with several individual devices in it with only one IP-address in the public address space, the customer is able to act as an ISP of their own, possibly without permission. By determining how many hosts that are in a network the ISP is able to suspect inappropriate sharing and also to gain more knowledge of how their service are used.

Procera Networks [19] is a business company that offers several solutions to network operators. They are able to structure, monitor, analyze and use the data for the operators, in order to increase the quality of their service. They are able to classify the Internet traffic with the help of their Deep Packet Inspection (DPI) engine. DPI is a form of network packet examiner which has been in longtime use in network management for monitoring and classifying of data directly from network

traffic flows [23]. With the help of Procera's DPI engine real network traffic flows was collected for this study and analyzed in order to evaluate different methods for NAT host identification.

1.2 Goal

Given the problem formulation, this study will focus on detecting NAT and the number of hosts for Windows computers by analyzing Internet traffic flows. The host identification process will be based on the behavior and analysis of the Internet traffic from anonymous hosts.

By analyzing specific flows for each IP-address, inferences will be made if an IP-address might be a NAT device and the number of hosts in the network. There exists several different methods that have been able to determine the number of hosts behind a NAT. One of the first NAT host detection methods were conducted by M. Bellovin [3] in 2002. Several other studies have also been conducted, which shows promising results for determining the number of hosts behind a NAT [22], [12]. The analysis will focus on the identification of larger private networks, numbered in 20 - 100 hosts, where the amount of flows each NAT device receives should be much larger than for a single host. The specific flows that will be used for the identification will be selected based on expert knowledge provided by Procera Networks.

This thesis analyzes datasets from two different sources. The first data source is used to analyze Windows Update flows from three VM's with different Windows Operating Systems (OS). This was done in order to provide ground truth data that will be used for the detection of different hosts behind a NAT.

The other datasets are provided by Procera Networks and contain real network traffic. Different methods are derived in this thesis in order to analyze the traffic flows.

1.3 Disposition

Chapter 2 provides background information regarding IP, TCP, NAT and the different tools used in this study. The chapter will explain what a NAT is, how it works and what different types of NAT that exists. It will also provide background information regarding TCP and IP header fields which is important information for this study. The previous studies that have been researched are presented in Chapter 3. Several methods are explained for different approaches which can be used to determine the number of hosts behind a NAT.

The different datasets and the specific flows used in the detection methods will be explained in Chapter 4.

Chapter 5 presents the research and results from an empirical study made on Windows Update flows from three different Windows OSes. The different methods that are used to evaluate and analyze the Internet traffic flows for this study on the datasets provided by Procera Networks are summarized in Chapter 6. Chapter 7 presents the results and evaluations for the datasets that were used for NAT detection and host counting.

The summary and conclusions are presented in Chapter 8, as well as the future work aiming to increase the accuracy of the host counting methods.

2 Background

This chapter will give an introduction to IPv4, TCP and NAT which will be referred to later in the study. Information regarding what they are used for and how they work will be explained. The various tools that are used in the study will also be mentioned here.

2.1 TCP and IP header Information

This section contain information about the TCP, IP header and their fields. This is done in order to get a better understanding of the fields that are affected by a NAT and to introduce them as reference for further explanations. By examining the fields in the headers they can be used to detect NAT and to determine the number of hosts behind a NAT device. Therefore it is important to know what the header fields are used for.

IPv4

Internet Protocol version 4 (IPv4) [25] is an Internet protocol that is used in order to transmit packets, also referred to as datagrams, between hosts in a network. It is a connectionless protocol with a limited scope of functions. In order to provide reliability with the delivery of datagrams, it takes help of an upper laying transport protocol such as TCP or UDP. Today it is one of the core protocols that routes traffic on the Internet. IP is used for two major tasks, addressing and fragmentation. IP delivers datagrams from a source to a destination host solely by using addresses in its header. It also fragments and reassembles datagrams with the help of the fields in the header. The header and fields can be seen in Figure 1. Each datagram consists of an IP header and the data that is transported. Each header consists of 13 mandatory fields and one optional.

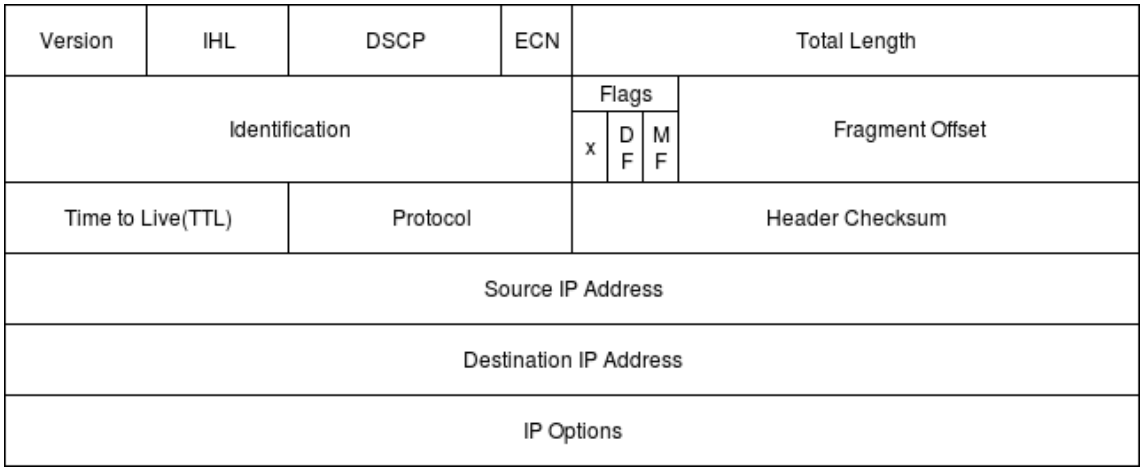

Figure 1: IPv4 Header.

The Version field contains which IP version in use, either IPv4 or IPv6.

The Internet Header Length (IHL) contains the length of the IP header in 32 bits words, ranging from value 5 to 15.

The Differentiated Services Code Point (DSCP) field is used to classify and manage network traffic in order to provide a quality of service.

Explicit Congestion Notification (ECN) is used in order to provide end to end notification of network congestion.

Total Length contains the value of the total length of the IP datagram in bytes, which includes the header and the data.

The Identification field which will be referred to as IPid in this report, it is used to identify the group of fragments for a single datagram. This field is used to reassemble the correct fragments at the destination.

There are three flags in the IP header which are each one bit big. The X flag is reserved and must be zero. Don't Fragment (DF) is used to indicate if the segment is allowed to be fragmented or not. If the More Fragments (MF) flag is zero it indicates that it is the last fragment. If MF is set to one more fragments will come.

Fragment Offset is a 13-bit long field which indicates the position in the datagram of the fragment.

Time to Live (TTL) is an eight-bit field which determines the maximum time a datagram may remain on the Internet. It contains a value that will decrement by one for each router that processes the datagram. When the value reaches zero the datagram will be dropped. The default TTL values may be different for different OSes, which can be used for NAT detection. Methods for this kind of NAT detection will be further explained in section 3.

The Protocol field defines which transport protocol that will be used in the data section of the datagram, where for example the value six is used for TCP and 17 for UDP.

The Header Checksum is a 16-bit field that is used for error detection of the header. When a datagram arrives at a router the checksum will be recalculated. If the recalculated value does not match the value in the checksum field the datagram will be dropped.

Source IP-address is a 32-bit sized field that contains the IPv4 address of the sender which is changed when the datagram passes through a NAT.

Destination IP-address is a 32-bit sized field that contains the IPv4 address of the destination. This field may also be altered when the datagrams pass through a NAT.

The IP Options consist of one or more 32-bit optional fields that do not have to be present if no IP options are used.

TCP

The Transmission Control Protocol (TCP) [26] is a common transport layer protocol used for reliable host to host communication that sits just above the Internet protocol. It is used to complement IP to transfer a stream of bytes between applications on IP networks.

TCP divides the data from the data stream into chunks and adds a TCP header to the data in order to create a TCP segment. The TCP segment contains a header and data section and is forwarded to the IP layer to be sent as Internet datagrams. The header follows the IP header and is used to supply TCP specific information. It contains ten required fields and one optional field as can be seen in Figure 2. The data section contains the data payload carried for the application.

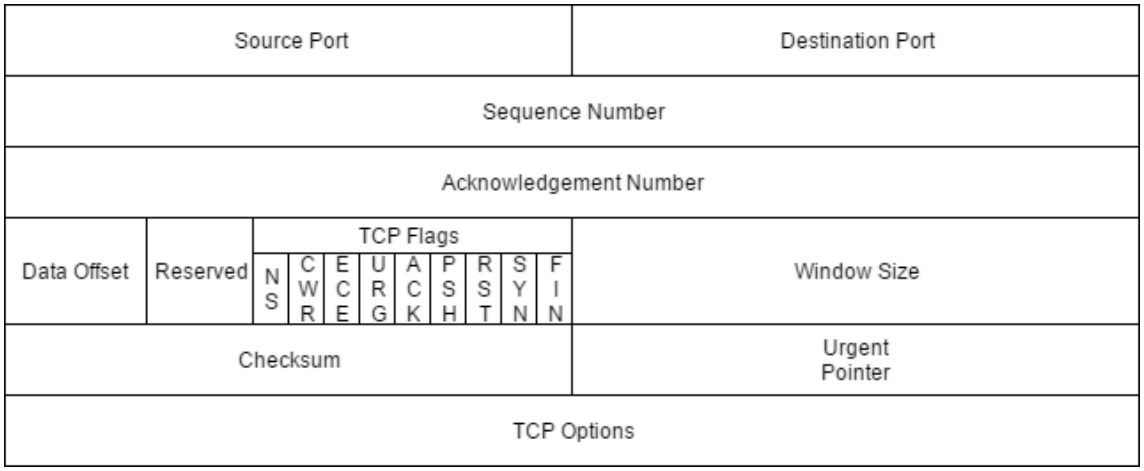

Figure 2: TCP Header.

Source Port is a 16-bit field containing the host's source port, which may be altered if the datagram travels through an (Network Address and Port Translation) NAPT.

Destination Port is a 16-bit field containing the host's destination port. This field may also be altered when traveling through an NAPT.

The Sequence Number field contains the initial sequence number generated by the OS if the SYN flag is set. The corresponding Acknowledgment number will be this number plus one. Otherwise, if the SYN flag is not set, this field will contain the sequence number of the first data byte in the packet.

The Acknowledgement Number field contains the value of the next sequence number of the TCP segment that the sender is expecting if the ACK flag is set.

Data Offset is a four-bit sized field that contains the total size of the TCP header in 32-bit words.

The Reserved field is only for future use and must be set to zero.

TCP contains six standard and three extended control flags that are one bit each to represent on and off. They are used to manage the data flow in specific situations. The SYN flag, for example, is used to initiate a connection, only the first packet in each direction should contain this flag. While the ACK flag acknowledges that the receiving data is valid, the FIN flag indicates that the connection is being shut down. Some of these flags can be used for NAT detection.

The Window Size is a 16-bit sized field that is used to regulate the amount of data in bytes that the sender of the packet is willing to receive.

The Checksum is a 16-bit field that is used to detect errors in the header and the data to let the receiver know if the data in the TCP segment have been altered. A pseudo header is also calculated for the checksum which contains the Source Address, the Destination Address, the Protocol, and TCP length carried in the IP header. The pseudo header adds up to a 96-bit header and the checksum is then calculated over the pseudo header as well as the TCP segment. This value is then placed in the TCP checksum field. This is used in order to deliver the segment to the right address and to verify that the data is error-free.

The Urgent Pointer field is used when the URG flag is set, in order to indicate that a segment of data must be delivered immediately. The 16-bit field will point to the position of the packet that holds the end of the urgent data.

2.2 Network Address Translation

The fundamental role of Network Address Translation (NAT) is to change address information in the Internet Protocol (IP) header of network packets [24], [29]. It was originally created as a solution to combat the shortage of IP-addresses provided by Internet Protocol Version 4 (IPv4). IPv4 does not contain enough IP-addresses in order for each device to have their own unique IP-address on the Internet. To solve this problem the Internet Assigned Number Authority (IANA) provided a

range of private IP-addresses that for example, a company can use in their private network. They have provided three blocks of private IP-address spaces which can be used without coordinating with IANA or any other Internet registry. But if an organization who uses a private address want to communicate with a public network the address have to change because several organizations can use these address spaces which makes routing impossible. The three blocks of IP-address that can be used for private use is ranged from 10.0.0.0 - 10.255.255.255, 172.16.0.0 - 172.31.255.255, 192.168.0.0 - 192.168.255.255.

NAT is used in order to map private IP-addresses which anyone can use to the more scarce public IP-addresses. The private network can consist of a number of hosts, which are connected to an NAT-Gateway. The NAT modifies or translates the private IP-addresses from the hosts to provide a connection to the public network, most often the Internet. The only IP-addresses visible in the public domain will be the public IP-addresses provided from an Internet Service Provider (ISP). This makes it possible to change the public IP-addresses without changing the private ones. The users are also able to change ISP without changing the addresses on the local network. Because a NAT device modifies the fields in the IP-header, hosts in the private domain are not visible for the hosts in the public domain. Therefore companies and organizations might employ NAT for security purposes in order to keep internal IP-addresses unreachable from external networks.

The NAT sits between the private and the public network and translates the private IP-address to a public one. This is done by creating bindings between addresses. In the next sections, the NAT types and the different binding methods will be explained.

Static NAT

Static NAT is the simplest form of a NAT router and works by mapping one IP-address in the private network space to one in the public network space. Each of the private hosts has a single public IP-address mapped to their NAT IP-address. The main reason to use a Static NAT is to hide the real IP-addresses but to allow certain network resources to be accessible via a certain IP-address. For example, if some customer has to access certain services from a database inside a private network. As can be seen in Figure 3 the hosts private IP-addresses are each mapped to their own public IP-address but the port numbers remain unchanged.

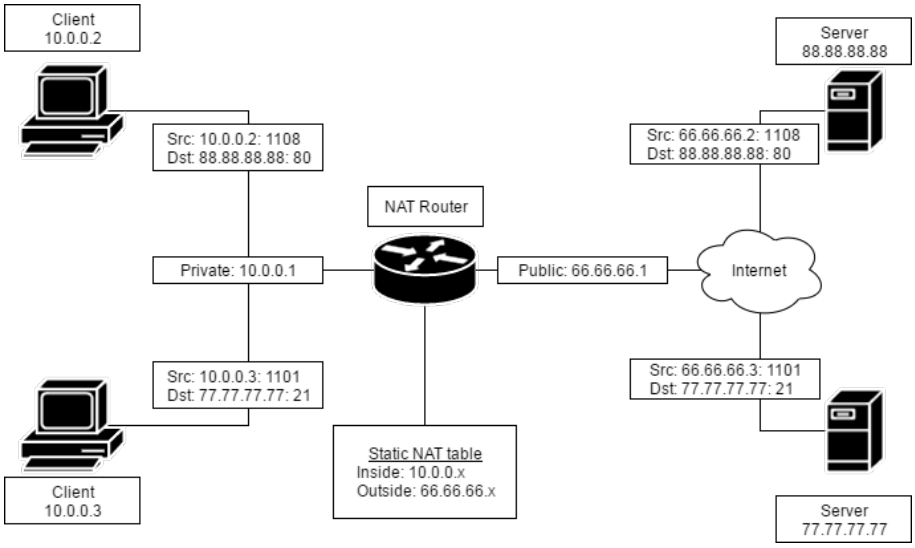

Figure 3: Static NAT.

Dynamic NAT

Dynamic NAT maps a private IP-address to a public IP-address chosen from a pool of public IP-addresses. The public IP-addresses are stored in NAT tables which are shared between all the hosts in the private network. When a private host initializes a connection the NAT chooses a currently unused IP-address from the pool. The host can be reached as long as the connection lives. When the connection is terminated the binding expires and the public IP is returned to the pool and can be reused for another host. As can be seen in Figure 4 the NAT router changes the private IP-addresses of a packet to a public IP-address chosen from a pool of available addresses in the Dynamic NAT table. When the packet returns the destination IP is changed back to the original private IP-address of the client.

This type of NAT is mostly used for larger organizations where several of the private network hosts have to communicate with hosts on a public network or vice versa. This makes the method a bit more complex in comparison to static NAT but increases the security because the mapping change which makes it hard to target a single host.

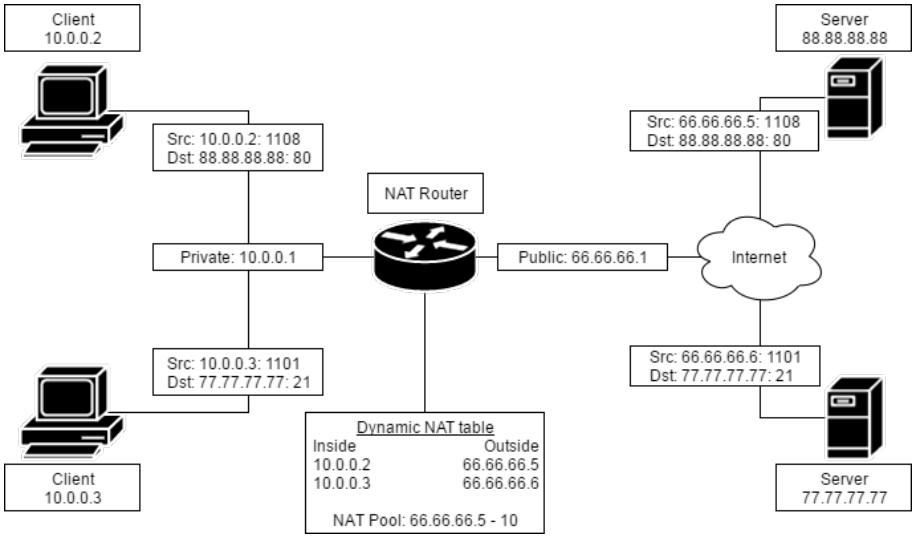

Figure 4: Dynamic NAT.

NAT Overload / NAPT

NAT overload or Network Address Port Translation (NAPT) can be seen as a combination of static and dynamic NAT but with the added functionality of Port Address Translation (PAT). This is the most commonly used NAT configuration and is often used in homes or small office networks which enable several computers to connect to the Internet while only utilizing one IP-address.

Often when people talk about NAT it is actually NAPT they refer to. NAPT is referred to by several different terms created by different groups, some of them are PAT, NAT overload, many-to-one NAT and IP masquerading. IP masquerading is often used to describe the security measure of hiding private IP-addresses behind a single public IP-address.

PAT is a technique that is used to translate port numbers. In Figure 5 two clients are represented by a single public IP-address but are differentiated by the port number as can be seen in the NAPT table. This allows several hosts to share one single IP-address. All the packets that leave the NAT gateway contain the same NAT IP-address but have different source port numbers. When a host inside the local network sends a packet to an address on the Internet, it goes through the NAT gateway which will store the host's IP-address and port number in the translation table and replace it with the global IP-address and port number. The reply packet will arrive at the public NAT IP-address and the IP-address and port number will be replaced back using the NAPT table in order to be forwarded to the correct private host IP-address in the local network. If a client wants to initialize a connection the port number should be chosen from the range 1 024 - 65 535 as is recommended according to RFC 2663[29].

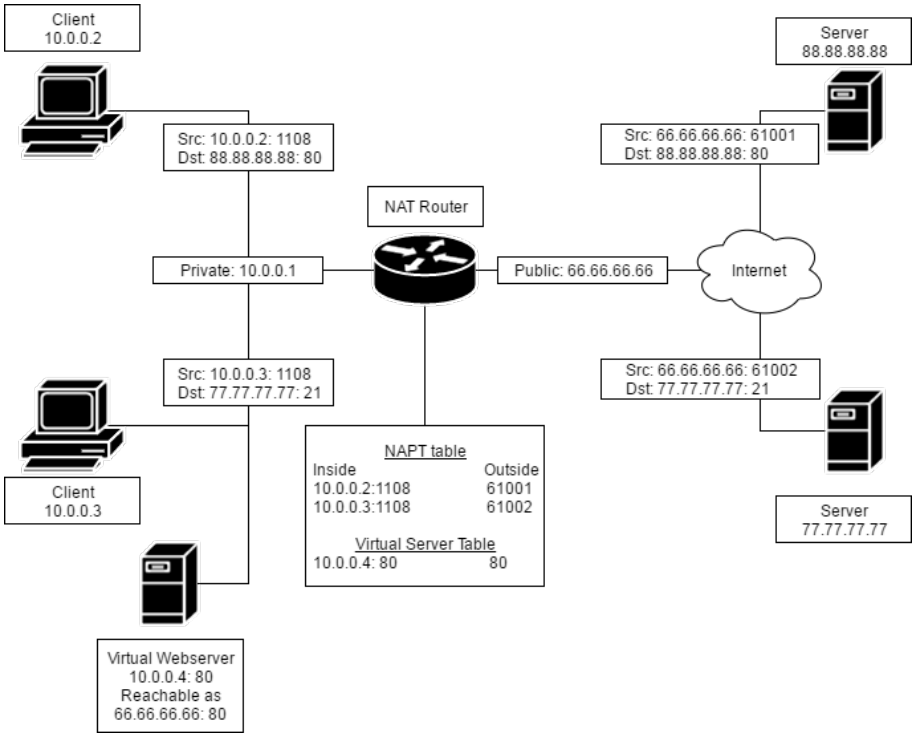

Figure 5: NAPT.

Changes Made by NAT

When a packet traverses through a NAT, several fields are changed. The most obvious changed fields are the source address of outgoing packets and destination address of incoming packets in the IP packet header. When the IP-addresses are changed the IP checksum have to be recomputed. This could otherwise cause the packet to be dropped because the receiver would believe that the packet has been corrupted. This also applies for TCP and UDP which are the most common transport protocols used for Internet traffic. They have a checksum for all of their data as well as a pseudo-checksum header that contain the source and destination address. Therefore when there are a change in the address the checksum have to be recalculated based on the translated IP-addresses and not the original ones. The source and destination port numbers for TCP and UDP are also changed when a packet is transferred through an NAPT.

The Time to Live (TTL) is a field in the IP-packet header that is affected by a NAT device in a network. The field is decremented by one for each router the packet goes through. Because of this, and the knowledge of original TTL values for different OS's, this field can been used for NAT detection.

Problems Caused by NAT

Unfortunately, NAT may cause some problems in certain circumstances where for example if the TCP header is encrypted the checksum field will not be able to get translated and the packet might be dropped. Some application protocols also carry an IP-address that might have to be translated. In order to do this, an Application-Level Gateway (ALG) is needed.

There is also problems where unauthorized NAT devices may provide unrestricted access to for example a private company network, thereby causing a significant security threat [28].

Another case is when NAT is used in order to distribute unauthorized Internet access to more hosts than allowed for from an ISP without their knowledge of it.

2.3 Tools

This section explain the libraries, softwares and tools used in the study.

Python

Python[9] is the chosen programming language for this study. It is a popular object-oriented high-level programming language. It is consistently designed, easy to read and have a minimal syntax which makes it easy to learn and shorten the development time. It was chosen because it contains a large library that can be used for scientific computing, more specific the modules for machine learning and handling of large datasets. It does have some disadvantages in that the execution can be slower compared to other imperative languages, but the advantages are believed to outweigh the disadvantages.

Jupyter Notebook

Jupyter Notebook[11] which was formerly known as IPython, is a web application environment that can be used for Python. It contains a cell-based environment where the project's code were created. The cells are able to contain code, plots, and text, which enabled the code to be organized in a structured way. This is very helpful when working with several calculations and understanding large datasets.

NumPy

NumPy[7] is a package for Python which contains several functions for scientific calculations. In this study, NumPy was used to handle large arrays and matrices with higher dimensions. Operations could be performed on these datasets with the provided functionalities from NumPy.

Pandas

Pandas[21] is a package for Python that is used to create data structures with high performance and provides tools to easily analyze the data. In this study, it was used to categorize and label the collected data into data structures to analyze it.

2.4 Chapter Summary

This chapter explained and gave an introduction to IP, TCP and NAT which will be refereed to in upcoming parts of the study. The TCP and IP section explain how TCP and IP work as well as explaining what the different fields in the headers are used for. The NAT section explains how different types of NAT works and what it is used for. The Tools section explains the various tools that have been used in the evaluations.

3 Related Studies

NAT host identification is used in order to determine the number of hosts that are connected behind a NAT device. This can be very valuable for network managing purposes in order to get information on how many users and devices that are connected to the network. There are several studies that examine different methods in order to identify the number of hosts behind the NAT. In this chapter, some of the methods will be explained in order to gain a basic knowledge of how others have performed NAT host detection.

3.1 Signature Based Detection Methods

A technique for counting NATted hosts, Bellovin 2002

Steven M. Bellovin[3] wrote a paper which according to him contained the first technique in order to determine the number of hosts behind a NAT device. The technique was based on examining the IP headers identification field. It was observed that the IPid acted as a simple counter for the packets of many operating systems. Consecutive packets emitted from a host would carry sequential IPid fields. By counting the number of consecutive packets grouped as strings from an IP-address it is possible to determine the number of hosts behind a NAT.

The IPid field is used to order fragment packets. It has to be unique in all fragmented packets that have the same protocol number, source and destination address. This is done in order to allow fragmentation and reassembly for a packet.

The algorithm that was employed by Bellovin was made to build sets of IPid sequences. When new a packet was received the algorithm scanned for the sets of sequences in order to find the best match. If a match was found the IPid was added to the sequence or else a new sequence was created.

A perfect match for the sequence and a new IPid was considered if the IPid was exactly one higher than the last received packet within the reasonable gap and time bounds.

The tests were performed by using real packet traces where the machines were grouped together in order to behave as if they were behind a real NAT. The subnet used was from their organization's wireless LAN, which contained hundreds of computers. The output of the algorithm was compared to the real IP-address data. It showed good results, where the algorithm was able to identify nearly all hosts.

However the algorithm had some problem when there were collisions in the IPid space. This could cause the algorithm to wrongly identify the number of hosts. The algorithm does only look for IPids that increase sequentially which only makes it

applicable for operating systems and packets that do so. Another limitation is that the IPid field does not have to behave as a counter, only that it is unique which makes this method invalid. It is also possible that the Don't Fragment bit in the IP header is not set which would not set the IPid field. Some NAT devices also resets the IPid to zero.

Bellovin conducted that the technique was best used for smaller networks behind a NAT, where the data can be collected by an Internet Service Provider (ISP) for analyzing the number of hosts in homes and small business networks.

Passive detection of NAT routers and client counting, Straka 2006

Kenneth Straka and Gavin Manes[34] wrote another NAT detection method and the limits on some already existing methods. Bellovin wrote that NAT detection using the IPid field can be somewhat unreliable to use for detecting different hosts, since it does not have to behave as a counter. Their solution was to combine the IPid field and the TTL values extracted from the IP packet header. Most OS's have default TTL values and can, therefore, be easily distinguished from each other. The TTL value is also decremented by one for each router the packet pass through and by examining the value of the TTL it can indicate if a NAT exists in the network. Unfortunately, this method still has some limitations since a client can change the default TTL values. Together with Bellovin's method of grouping sequential IPid's together they were able to gain a more accurate estimation of the number of clients in a network. In order to improve their method, they choose to also look at application level packet patterns, where they gave an example of how the detection would work for bursts of Post Office Protocol (POP) traffic. If the packets were found within close intervals between each other it could be an indication of several hosts who checks their e-mails. The usernames submitted to the POP-server could also be used to separate hosts. This method of identification can also be used on operating system updates such as Windows update and application updates such as antivirus updates. Even though they did not perform any experiments on a dataset, these types of host detections have been used in other research as will be seen below.

Application presence fingerprinting for NAT-aware router, Bi 2006

Bi et al.[4] conducted a new fingerprinting method based on application layer data in order to discover NAT. This detection method was introduced because it is hard for NAT routers to modify application layer data in order to avoid detection and that the application developers don't design features to avoid NAT detection. Usually,

one host has only one instance of an application, this fact can be used for host separation. In their research, they used Instant Messaging (IM) applications to distinguish hosts. Some IM applications only allow one instance to be active on a desktop, they have a large user population and the applications are usually active for long periods. They found that the IP-address of IM servers, TCP/UDP port numbers, and certain IM packets have some characteristics that can be used for fingerprinting.

Their algorithm checks each IP-address where IM packets come from. It checks the destination IP-address and source port number in order to identify if the packets belong to an existing channel between the client and the application server or otherwise it will create a new channel. The algorithm then counts the number of channels between the given IP-address and the IM application in order to make a verdict. If the number of channels is equal or greater than the maximum of allowed channels for each IM application it is concluded that the IP-address is a NAT gateway address. The algorithm then removes each expired channel given a maximum idle time.

Their experiment was conducted on a campus network and on the China Education and Research Network (CERNET), which is a network used for education and research. The IMs they used was MSN Messenger and Google Talk. Even though they mention that they performed an experiment, no results are presented to show the accuracy of the algorithm.

NAT usage in residential broadband networks, Maier 2011

Maier et al.[14] used an approach to detect NAT and estimate the number of hosts behind it by using the TTL field in the IP header and HTTP user-agent strings. Their approach used the knowledge from previous studies regarding TCP/IP fingerprinting in order to come up with a more reliable NAT detection method.

Their dataset are based on packet-level observations in Digital Subscriber Lines (DSL) connections from a large ISP. They were able to monitor 20,000 anonymized DSL.

In order to detect if NAT was used on a DSL line, they used the fact that OS's have different TTL values on outgoing packets. Windows have an initial TTL value of 128, MacOS and Linux have 64. For every router that the packet travels through the TTL value will decrement by one. They did also know that the hop distance from the customer's equipment to their monitor point was one. They used this knowledge to determine if there was a NAT in the customer's network. To count the number of hosts behind a NAT on the DSL line they counted the number of

TTL observations from each line to distinguish different OSes. HTTP user agent strings of regular browsers contain information about OS and browser versions. By combining the TTL values and observing the OS and the browser versions they were able to identify the number of different hosts behind a NAT. They made the conclusion that it was unlikely for several customers to have both the same OS and browser family versions.

In their results, more than 90% of the lines they observed used NAT and they also found that 30-50% of the lines had more than one active host. In order to see if multiple hosts on one line were active at the same time they computed a minimal interactivity time, which showed that 10% of the DSL lines had more than one host active at the same time. They were not able to distinguish between hosts with identical OS and browser. It is also possible that they wrongly classify a computer that has two OS's as two hosts, or if a user updates his browser during the observation period. If a NAT gateway does not decrement the TTL value it will not be classified as a NAT. This method will also be unable to detect hosts based on the HTTP knowledge if the flows are encrypted, which will hide the user agent information.

Counting NATted hosts by observing TCP/IP field behaviors, Mongkolluksamee 2012

Mongkolluksamee et al.[18] developed a technique to study long-term NAT trends from 2001-2010 using traces from the Measurement and Analysis on the WIDE Internet (MAWI) group. In their study, they examined the behavior of IPid, TCP sequence number and source port to identify different operating systems. By examining the behavior of these three fields they were able to identify patterns of different operating systems.

They extended the work of Bellovin's method in order to estimate the number of NAT hosts from network traffic, where they included TCP source port and sequence number. In contrast to Maier that looked on HTTP user agent strings to identify different OS, they discovered that each OS has its own way of selecting a starting TCP sequence number and a TCP source port for each connection. They added this to Bellovin's method to identify different host when the IPid is in a per flow or random manner.

The method they used is divided into two part. In the first part, they collect IPid sequences, TCP sequences, and TCP source ports and then they ordered the sequences in an increasing order. The TCP sequence is created by calculating the arrival time of the packet and gap limit in the sequence. Based on the results from

the difference of the previous packet and the arriving packet it is classified differently. The IPid sequence was calculated in the same way, but they had to create two sets for each new IPid which is added to the best matching sequence set. The parameters chosen in the algorithm to construct the sequences are based on table parameters from Bellovin.

When the sequence construction phase is done they start to classify hosts. This is done in three steps where they first try to associate TCP sequence numbers with IPid sequences. When a TCP sequence and IPid have been associated together they observe patterns of the TCP sequence starting number to distinguish different OSes from each other. Step two is when they only found one TCP sequence for each IPid sequence and tries to distinguish which type of IPid the TCP connection is associated with. The last part observes the TCP source port behavior, this is done if a TPC sequence cannot be associated with any IPid sequence and therefore they try to discover patterns by looking on to the source ports.

The method was evaluated by using two trace files, one from a synthetic NAT and one from real NAT traffic collected from a wireless router. The synthetic NAT traffic was collected from 16 hosts with different OSes. By using this method they found 18 estimated hosts, with 15 true positives and three false positives. This result was slightly better when they compared it with Bellovin's method. Their method was able to identify Windows, FreeBSD and MacOS with a high accuracy but had problems with detecting OpenBSD, where the algorithm would count several OpenBSD hosts as one host.

When evaluating the results from the real NAT traffic they compared their results with pOf which is a passive OS fingerprinting tool, which gave similar results.

They used their method to study the long-term NAT trends on the data from the MAWI traces from 2001 - 2010. Their results showed that around 2% of the IP-addresses were NATed and on average there were five hosts behind every NAT.

A hybrid packet clustering approach for NAT host analysis, Zhang 2015

Zhang et al.[36] also performed a study in order to discover how many hosts that were located behind a NAT. They used a method where they gathered HTTP data and extracted cookie ID, application ID and user-agent information from each packet group in order to gain knowledge of which IDs belongs to which host. They created an environment where the network traffic was collected inside two laboratory networks before a NAT device so all IP-addresses were visible. Their method would work on the outside of the NAT network, but the traffic captured before the NAT gateway was used to validate the results. A sliding

time window technique in combination with cookie ID verification was used to verify when two HTTP's requests belong to the same host. They analyzed the HTTP header to employ the rules in each time window. If two HTTP requests were sent to the same destination within in a small time period they made the conclusion that they are probably from the same host. They also checked if the User-agent was the same for each HTTP header. A cookie cluster was used which contained cookies that would not change in short periods of time like session cookies. This cookie cluster was combined with a cookie ID cluster in order to verify if they were from the same host. Cookie ID's were used to verify and connect two HTTP requests from the sliding time window cluster or the cookies cluster to see if there was any cookie ID conflict between them. They created a cookie ID table that contained cookie IDs from different websites to check the connections. Their results revealed that from their NAT networks with around 100 hosts, they had an average accuracy of more than 90% and a coverage of more than 50%. However, this method will not work if the HTTP header is encrypted.

Identification of hosts behind a NAT device utilizing multiple fields of IP and TCP, Park 2016

In a study made by Park H et al.[22] they tried to identify the number of hosts behind a NAT by utilizing multiple IP and TCP header fields. They used the IPid, TTL, SYN, source port and timestamp fields in order to separate the hosts. Their method to identify the number of hosts was made in two phases. The first phase was to determine what OS the packets were directed to. The algorithm examined if the protocol in use was TCP and that the SYN flag was used. In order to determine the OS a table with TTL initial values was used, similar to Maier's research[14].

The next phase separated the individual hosts, they did this with two methods. One of the methods was by using IPid and source port number and the other were a method based on TCP timestamp. They generated a host list which contained host information, if a packet came from a new host a host list item was created and added to the list. If the packet came from a already existing host then the packet items were added to the appropriate host list item. In order to determine if a packet came from the same host, they used a table that contained already determined parameter values such as the difference in packet arrival time, IPid value, source port number and the number of items in the list. If the difference between the arriving packet and the last packet in the list were in range according to the parameter table, as well as if the number of items in the list was greater than the value in the parameter table, the packets were considered to come from the

same host. Since Windows have sequential source port numbers in the TCP SYN packets and IPid they were used to identify Windows hosts. The TCP timestamp method can be used to identify the hosts by identifying the pattern of a linear equation, which depends on the OS, the initial timestamp and the current time. They performed the test by using two OS's and in a local small-scale environment. They were able to gain a result with an accuracy ranging from 71 - 100%, precision of 83 - 100% and sensitivity of 83 - 100%.The results were considerably better compared to only using IPid like Bellovin or just using user-agent strings.

3.2 Behavior Based Detection Methods

Remote NAT detect algorithm based on support vector machine, Rui 2009

Based on previous research, Rui et al.[13] proposed a new method for remote NAT host detection that does not depend on any special fields in the packet header. The technique uses a Support Vector Machine (SVM) method to analyze the network traffic from a NAT together with a remote NAT detection algorithm.

They purposed a method to detect NAT devices passively by analyzing network traffic based on eight features. The chosen features can be seen in Table 1.

Table 1: The 8 features for network traffic analyzing.

	Features
1	The number of packets sent out
2	The number of packets received
3	The number of UDP packets
4	The number of TCP packets
5	The number of DNS request packets
6	The number of FIN packets
7	The number of RST packets
8	The number of SYN packets

The features were chosen based on the behavior of the NAT network traffic. A NAT network will send out more bytes, connections, DNS requests, protocols and have a more complex behavior than regular host traffic. These features are used to represent the network traffic from a NAT and were filtered from data traces. The network traffic is represented as eight dimension serial vectors during a set duration.

They discovered that sometimes the network traffic from hosts behind a NAT and ordinary hosts does not differ that much, for example when all the hosts behind

a NAT are inactive. This makes it harder to see the difference between a NAT host and an ordinary host. In order to gain a higher accuracy for their method, they purposed a function to filter out the inactive hosts form the network traffic leaving only the active ones left in the dataset.

After the network traffic had been filtered, the SVM was used to analyze the vectors. The vectors were evaluated as a binary classification problem where the SVM analyzed if the data were labeled to come from an ordinary host or a NAT host. The network traffic was captured from five networks. Four of the networks were placed behind a NAT and the number of hosts ranged from two - five. One network only contained one host and was not placed behind a NAT. The results showed that as the number of hosts behind the NAT device increased so did the accuracy and the specification. When the number of hosts reached five the accuracy and specificity reached nearly 100% with sensitivity reaching 80%.

Passive Remote Source NAT Detection Using Behavior Statistics Derived from Netflow, Dietz 2013

Dietz et al.[1] proposed a passive NAT detection method in order to detect rogue NAT devices. They did this by employing machine learning algorithms based on behavior statics from NetFlow data. The machine learning algorithms used where a Support Vector Machine (SVM) and a C4.5 decision tree algorithm. Their method is based on the assumption that NAT traffic will behave differently than network traffic with only one host. In order to model the NAT behavior, they used nine features which they extracted from the NetFlows records.

The NetFlow data was collected from an ISP during eight days. Most of the traffic belonged to DNS traffic. This counted up to a total of 6 631 383 anonymize records with labeled NAT and non-NAT traffic.

The method is divided into two steps where they first train the machine learning algorithm with the data from a NAT which is based on the features from the NetFlow data. The trained classifier was then fed with unlabeled feature vectors to classify NAT and non-NAT traffic.

Using several NetFlow records they computed feature vectors based on each unique IP-address found in the records which started during a chosen time window. The result from this is a set of feature vectors that is based on each unique IP-address.

The features derived from NetFlow are visible in Table 2.

Table 2: Top NetFlow features based on unique IP-addresses.

	Features
1	The number of TCP NetFlow records
2	The number of UDP NetFlow records
3	The number of NetFlow records belonging to DNS
4	The number of NetFlow records belonging to SMTP
5	The number of NetFlow records belonging to Email traffic
6	The number of NetFlow records with SYN flag set
7	The number of NetFlow records with RST flag set
8	The number of bytes exchanged within a flow
9	The number of packets transmitted within a specific flow

The machine learning algorithms were then fed with training and testing sets from the feature vectors. In order to not introduce bias when applying the machine learning algorithms, they derived a balanced dataset by randomly sample the feature vectors on the NAT class. This was done because an imbalance was detected between the nonNAT traffic and the feature vectors created. The number of feature vectors created was rather small in comparison to the amount of nonNAT traffic.

The results showed that the C4.5 algorithm had an accuracy of 95.35% on the unbalanced data and 89.35% on the balanced data. The SVM had an accuracy of 95.10% on the unbalanced data and 81.29% on the balanced data. This reveled that the classifier presented better results when it was trained with biased data. They also conducted that C4.5 performed better and faster compared to the SVM. In their finding, they describe that they are not yet sure if the unbalanced dataset should be classified as bias or if it could be used as a feature to discover NAT. In their experiments, they also found two phenomenon's that could be used for NAT host identification. The first one was that source port translation done by the NAT gateway on outgoing packets were made in a deterministic way according to programmatic conditions. This was not further investigated but saved as future work. The second phenomenon was that the SYN packet sizes extracted from the NetFlow records had different packet sizes based on their OS, which might be used to identify the number of different operating system behind a NAT.

They have also released a master thesis where they expanded their classifier to include the use of source port sequences and average SYN-flow byte sizes, as well as the user behavior which is based on 11 features instead for the NAT detection[8]. The SYN flows which the SYN-based detection is based on, appeared seldom which resulted in a low detection rate for this approach. 0.22% of the total number of flows were classified as SYN flows and only a subset of those could be used for detecting

NAT, but they were able to determine different OSes from those flows. The number of source port sequences found in the dataset was also low because some flows which could be part of a sequence was lost, because of the time frame of the flow exports were set to five minutes by the provider, which resulted in a low detection rate.

Can we identify NAT behavior by analyzing Traffic Flows? Gokcen 2014

Gokcen et al.[10] identified behavior from a NAT by analyzing traffic flows using machine learning principles. They did not try to estimate the number of hosts behind a NAT but only wanted to discover the existence of NAT in the connection. The Maier et al [14] approach was re-implemented and compared to their own machine learning approach.

The datasets they used came from two different organizations, which included encrypted and non-encrypted traffic. One of the datasets was collected during a week on their own network and the other set was delivered to them through a private partner. They performed tests on both of the datasets and labeled all flows as NAT flows and OTHER flows. The number of flows used was 321 209.

From the TCP dump traffic, they computed the features for each flow. They used NetMate which is an open-source flow generator to generate flows and to retrieve statistical features from the traffic traces. They did not use IP-address and port numbers because they believed they might cause bias in the results. In their approach, they wanted to find patterns for the behavior of a NAT and did not use any application layer information.

The fingerprinting approach was evaluated in four steps based on methods tested by Maier et al[14]. They tested it by evaluating the TTL range, distinct TTL values for each IP, different OS information and browser information in the HTTP user agent strings.

This approach was then compared to their own proposed approach where they employed two machine learning techniques. They used a C4.5 decision tree classifier and a Naive Bayes probabilistic classifier. These two classifiers were then compared to determine which worked best on the features.

Their results showed that the passive fingerprint classifiers worked on certain NAT behaviors, but as the NAT behavior became more complex it was harder to gain accurate results. On the datasets, the best detection rate for the NAT flows were 100%, with a False positive rate of 6% and 2,7%.

The machine learning approach showed high-performance accuracy with a detection rate of 98-99% for C4.5 on the NAT and OTHER flows, with a low false positive rate of 2-4%. Naive Bayes was not as good and performed differently on the two data sets. Compared to the results from the fingerprint approach it can

be shown that the C4.5 classification worked best on the two datasets. The C4.5 algorithm was also able to identify the most important features to detect NAT behavior. These features are visible in Table 3.

Table 3: Most important features classified by C4.5.

	Features
1	The average number of bytes in a sub flow in the forwarded direction
2	Total bytes in backwards direction
3	Mean size of packets sent in the forwarded direction
4	Maximum duration the flows were active
5	The size of the smallest packet in the forwarded direction
6	The size of the biggest packet in the backwards direction
7	Standard deviation from the mean of the packet sent in the backwards direction

Passive NAT detection using HTTP access logs, Komarek 2016

In one of the most recent papers, Komarek et al.[12]proposed a NAT detection method using HTTP logs. Similar to Gokcen et al.[13] and Rui et al.[10] they looked at the behavior of a NAT to discover it with the help of a machine learning classifier. Their method is divided into two steps; in the first step, they analyzed the host behavior. It consists of feature extraction, statics collection, and window selection. In the second part, they trained a SVM in order to label IP-address as NAT or end host.

For each host that was identified in the HTTP logs they performed feature extraction based on eight features in Table 4

Table 4: Features extracted from HTTP logs

	Features
1	The number of unique contacted IP-addresses
2	The number of unique User-Agent strings
3	The number of unique OSes
4	The number of unique Internet browsers
5	The number of persistent connections
6	The number of uploaded bytes
7	The number of downloaded bytes
8	The number of sent HTTP requests

These features were collected to identify a NAT from an ordinary end host. They

were collected in time windows sequences with 30 minutes for each sequence covering 24 hours.

They gathered the HTTP access logs from four different corporate networks during two working days. The networks were selected to be of different sizes with 3 000, 5 000, 10 000 and 25 000 hosts. From these datasets, they generated an artificial NAT by joining HTTP logs together to simulate hosts behind NAT. The data was then labeled in two sets: Artificial NAT and end hosts to train the classifier.

To evaluate their NAT detection method they conducted an evaluation in five separate scenarios:

- Time-drift: The classifier is trained on the data captured from the first day and is then evaluated by the data captured from the second day. This is done to show that the classifier can operate on data from the same network in future days with a high accuracy.
- Cross-validation: In order to see if the classifier can be used on new network data.
- Hosts with the same OS and/or Internet browser: To test if their method were able to identify the host where both Beverly's[3] and Maier's[14] method would fail.
- Sensitivity to contaminated training sets: In order to measure the impact of the classifier if a real NAT was presented in the training set.
- Evaluation on the Network with real NATs: where they used HTTP logs from a corporate network to test their classifier, which contained 1 717 end hosts and 166 NAT devices.

All of the scenarios achieved an accuracy ranging from 93.91-99.36%, precision 89.21-98% and recall of 84.35-95%. The test on the Network with real NAT devices with the trained classifier achieved the highest score on all the measurement. It was also observed that NAT devices that contained five or more hosts were able to be detected in more than 96% or the times.

3.3 Chapter Summary

The different approaches described in this chapter can be categorized into two detection methods namely signature or behavior-oriented.

The signature based or fingerprinting approaches are detection methods which look for signatures in the IP, TCP and HTTP fields, such as the TTL, IPid,

User-agent strings, port numbers and more. By using these fields it is possible to gain information about the number of hosts by observing a number of similar sequences on the packets from an IP-address. It is also possible to detect OS's, browser versions and applications in use. By observing and making assumptions of how different applications work they can be used to identify different hosts from one IP-address. In order to increase the detection accuracy, some methods use different fields together. There are some problems associated with this approach where the fields associated with the detection might not be in use. For example, the IPid might not be set or behave in an unpredictable manner or the TTL values might not be set to the standard OS values. Some OSes creates the values in the header fields differently than others which make them harder to detect, because of this some hosts might avoid detection. Although there can be some problems by building a method that uses fields that are not originally created for host detection, most of the methods achieved a high accuracy during the tests.

The behavior-based approaches are methods that examine how the traffic from a NAT behaves. Papers that use these methods are written by Gokchen et al, Dietz et al, Rui et al and Komarek et al. By making the conclusion that traffic from a NAT behaves differently than an ordinary host they are able to identify NATed networks. Some of the most common features that are examined in these approaches are the number of different packets sent and received, and the number of bytes exchanged. A NAT usually sends out more traffic than a single ordinary host, because usually there are several hosts hidden behind the NAT gateway. Often the accuracy of these methods increases when the number of hosts in the NAT network was greater. Usually, these methods use machine learning methods in order to classify if a host is a NAT or not. They used different methods in order to first label the hosts and then train the classifier on the data. Unfortunately, the machine learning methods are only able to classify if a host is a NAT device or not, and is not able to determine the number of hosts in an NATed network. The detection methods were able to classify a NAT device with nearly 100% accuracy if the NAT network consisted of more than five hosts.

Unfortunately, some of these methods might not be viable when the headers are encrypted such as in HTTPS. The papers written by Komarek et al, Gokcen et al, Zhang et al and Maier et al might not work with encrypted fields because they used information from the HTTP header fields.

Those methods described in this section that are able to identify the number of hosts behind a NAT device are described in papers written by Bellovin, Straka et al, Bi et al, Maier et al, Mongkolluksamee et al, Zhang et al and Park H et al.

The next section will explain the datasets and the aspects used in the NAT host detection methods.

4 Datasets and Detection Aspects

This section will explain the data sources and datasets used in the study. It will explain the attributes of the different datasets, how they were collected and for how long. This section will also present the NAT host aspects that were used in the study. These aspects were used in the analysis in order to create an accurate NAT host detection model.

4.1 Data Sources

The datasets used in this study is gathered from real network traffic traces from two different sources. The first sources of data were provided by Procera Networks which collected the data with a Deep Packet Inspection (DPI) engine. It collects information from an ISP and inspects the header and the data of each packet that passes through the inspection point.

The second dataset source were gathered from Karlstad's University network. The datasets do only consists of traffic that is related to how Windows receives updates. This was done in order to gain ground-truth information of how the flows looks like when a Windows computer downloads the updates to create an accurate detection model. The datasets were collected by a Procera DPI appliance located between the Internet and three VMs running different versions of Windows. The traffic was processed and then transferred to an HDF file.

The gathered packets were grouped into traffic flows [5], which is a sequence of packets that travels between a source and destination. The packets grouped into flows will be identified based on a combination of source and destination address, port numbers and transport protocols. By grouping the packets together into flows it will minimize the amount of data in the datasets while still maintaining a good representation of the traffic.

4.2 Procera Network Datasets

Three different real network traffic datasets were provided by Procera Networks which are analyzed and used to detect different hosts behind a NAT based on the detection models. These datasets consisted of traffic gathered from several different IP-addresses, for different durations, and from different countries.

Small Cellular

The small cellular dataset was gathered by Procera's DPI device. It was collected during approximately three hours. The dataset consisted of 781 793 unique flows

with 12 features. It used anonymized IP-addresses and did not contain any information regarding if the hosts were labeled as a NAT device or a regular host. The dataset was used to gain knowledge of how the data looked like and to make the first attempts for detecting the hosts behind a NAT. Different applications flows were examined in order to detect the hosts. The flows used for the detection of the hosts can be seen in Table 5 and 6. These applications were searched for in the dataset but not all of them were present. The flows were created by grouping ten or more packets which had the same properties.

Large Cellular

This was a much larger dataset than the previous one and contained 42 488 795 flows. Like the first dataset, it was collected from a cellular network but for 18,7 hours. It was similar to the first dataset in the sense that it did not have any information regarding if NAT hosts existed in the dataset. All IP-addresses were anonymous and each flow had the same attributes as the first dataset except that the size of each flow in bytes was now present. The same measurements were performed on this dataset as with the first one, in addition to some new measurements. Another difference is that some of this datasets flows were grouped together differently. In the previous dataset, a flow was only created if it contained ten or more packets that had similar properties. In order to increase the presence of antivirus flows for this dataset they were grouped on as few as one packet. This was done for the applications present in Table 5 because these applications might send packets to check for new updates and those flows may consist of less than ten packets.

DSL and Cellular

This was the largest dataset provided by Procera and originally it consisted of 102 866 346 flows, however some flows were not classified correctly and had to be removed. The cleaned dataset instead consisted of 68 596 470 flows and the traffic was collected during approximately seven days. As with the previous datasets, it did not contain any ground truth regarding the presence of NAT routers and all the IP-addresses was anonymous. The dataset contained the same attributes as the Large Cellular dataset, and the same measurements were performed.

4.3 KAU Lab Datasets

Three Virtual Machines (VM's) with different Windows OSes was setup on a computer at Karlstad university. The VM's did only have Windows installed on them and the purpose was to receive Windows updates from Microsoft so the traffic

behavior could be analyzed. The VM's was not used for anything else while the traffic was collected. The conclusions drawn from the traffic behaviors are used as ground truth for the detection models and are used to detect different NAT hosts.

Windows 7 Dataset

This dataset was gathered for around 15 days during the same time period as the Windows 10 computer. This dataset contained 12 717 flows of which 82 were classified as Windows Update flows, which is 0.64% of all the flows. The majority of the flows in the dataset consisted of DNS and SSL v3 which represented 6 416 and 3 125 of the flows. In total 20 different services was observed for this dataset.

Windows 8.1 Dataset

The Windows 8.1 computer was connected and searched for updates for around 24 days. During this time the computer established 7 641 flows, where 147 of those were classified as Windows Update. It was revealed that the computer received Windows update flows constantly during the time it was connected to the Internet with around six flows per day. DNS was the service that received the majority of the flows and it consisted of 4 102 flows, in total 13 different services was found.

Windows 10 Dataset

The Windows 10 dataset was gathered around the same time as the Windows 7 VM and for the same duration which was around 15 days. This dataset contained 20 058 flows and 267 of them were classified as Windows Update flows, which is around 1.33%. The services that received the highest number of flows in this dataset was DNS with 8 891 flows and SSL v3 with 5 806 flows. 19 different services were found in the dataset where the majority of services were the same as in the previous datasets.

4.4 NAT Host Detection Aspects

The different application update flows examined and searched for in the datasets are listed in Table 5 and 6. The flows created by the applications listed in these tables will from here on be referenced to as NAT detection flows.

The antivirus software's presented in Table 5 was chosen based on how common they are and on the expert knowledge provided by Procera Networks. According to AV-comparatives security survey [2] of 2017 and OPSWAT market share report [20], the antiviruses listed in Table 5 are some of the most commonly used for protection.

For Windows users, it is very common to have an antivirus program installed in order to protect, detect and remove harmful software from computers. According to Microsoft Security Intelligence Report [6], 88% of all the users which have the Malicious Software Removal Tool (MSRT) installed and enabled to provide information to Microsoft, have a real-time security software installed on their computer. The antivirus software's do often receive a lot of updates from the companies responsible for them in order to keep them up to date against the most recent viruses. Therefore it is expected that several of the antiviruses presented in Table 5 will be found in the provided datasets.

The software's featured in Table 6 were chosen based on an expert knowledge of application updates behavior in networks. These applications commonly retrieve updates from their companies and are used by many Internet users.

Table 5: Antivirus Flow Labels.

Antivirus		
360 AntiVirus	Bitdefender QuickScan	NOD32 update
AVG	Bullguard update	ProxyAV update
AVG Anti-Virus update	ClamAV update	Rising Antivirus
ArcaVir Antivirus update	F-Prot Antivirus update	Sophos Anti-Virus update
Ashampoo AntiVirus update	F-Secure virus definition	Symantec Anti-Virus
Avast! antivirus update	Intego update	Symantec LiveUpdate
Avira AntiVir update	Kaspersky update	Trend Micro AntiVirus
Bitdefender Antivirus update	McAfee VirusScan update	eScan update

Table 6: Software-update Flow Labels.

Software-updates	
APT	Microsoft Auto Update
Adobe Update Manager	Microsoft BITS
Apple Software Update	Nintendo DSi network update
Image Packaging System	Tesla update
Java update	Windows Update
Wireshark update	

4.5 Chapter Summary

This chapter has explained the different datasets used in the study. The sources of the datasets as well their properties were presented.

The detection aspects that is used to detect and count the NAT hosts were presented in Section 4.4.

5 Windows Update Empirical Study

This study was performed to analyze Windows Update flows and to gain ground truth data to determine methods for counting the hosts. The Windows Update flows from three different OSes were examined because as previously mentioned this study focuses on the detection of hosts with a Windows OS installed. Since Microsoft releases Windows updates to the majority of their OSes on a frequent basis these flows can be used in the detection [35].

A Windows OS can be configured to download and install updates automatically which increases the chance that the users receives the updates [17]. From Windows Vista to newer Operating Systems it is possible to activate automatic updates through the control panel which will download and install new updates when they are available. Automatic download and install are both recommend settings for Windows and for Windows 10 these settings have been set to default.

According to a report made by Microsoft in 2008 [15] Windows update had around 500 million clients which processed 350 million scans per day. Around 90% of all the clients used automatic updates which proves that there exists a lot of automatic Windows Update traffic which can be collected and analyzed to distinguish different hosts.

5.1 Windows Update Evaluation Methods

This section describes the methods which were used to analyses the traffic flows from the three Windows OSes. This study focuses on analyzing the downloaded update flows in order to be able to distinguish updates which several different hosts could have received.

It was of interest to see how the flows were distributed over time to be able to detect patterns which could be used for NAT host detection. Therefore the first analysis were made on the downloaded byte size and arrival time of each flow. By analyzing the distribution of the flows for the three OSes common traits between them might be detected which could be an important key for the host detection.

The second method was performed on the time difference between a new flow and a previous flow. The method calculated the time difference between all the flows in the dataset and displayed the time difference frequencies. This analysis was done in order to get a better knowledge of how long time it is between the updates. If a common frequency in time difference can be found between the flows, this trait could be used to identify individual hosts.

The last analysis was to study how the size of the flows could affect the time differences between them. If flows of a certain size had the same time difference,

this information could possibly be used for the host detection.

5.2 Windows 7 Analysis

From the Windows 7 dataset 82 out of the total 12 717 flows were classified as Windows Update with different downloaded sizes. Statistics of the downloaded update sizes are presented in Table 7. The table shows that the majority of the flows were quite small in their size and a few flows were really big.

Table 7: Size statistics for downloaded Windows 8.1 Updates.

Statistics	**Download Size (Bytes)**
Mean	7 108 362
Standard Deviation	22 599 020
Minimum	667
25-Percentile	1 722
50-Percentile	2 532
75-Percentile	470 105
Maximum	108 654 248

In Figure 6 the byte size of each flow is displayed on the y-axis and the start time for each flow is presented on the x-axis. As can be seen in the figure, the computer received update flows every day. The majority of the flows were received alone but some flows were downloaded in bursts with as many as 13 flows during the same time interval. Most of the flows downloaded individually were of small sizes, with an average of 3 465 bytes and a standard deviation of 8 442 bytes. The largest individual flow were found to be 55 280 bytes big.

The sizes of the flows in the flow bursts were on average 11 812 210 bytes big, but with a standard deviation of 27 431 260 bytes which shows that the sizes of the flows in the bursts is varying a lot. The smallest flow found in the flow bursts was 667 bytes big and the largest flow was 108 654 248 bytes.

On average the computer received around 5.6 flows per day and the largest flow burst arrived the first day.

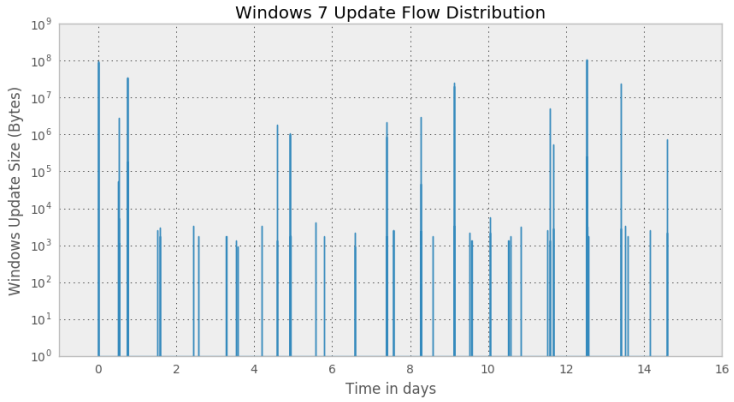

Figure 6: Distribution of Windows 7 Update flows with the start time against the amount of bytes they transfer.

Figure 7 shows the frequency of start times differences between two flows in order to see how long the duration is between each download. As can be observed more than half of the flows started directly after one other, these are the flows in the flow bursts. The rest of the flows had a start time difference that ranged from 9.7 minutes to at most 20.5 hours, where the average start time difference was 8.74 hours. The flows which has the longest time difference is the first flow in the flow bursts.

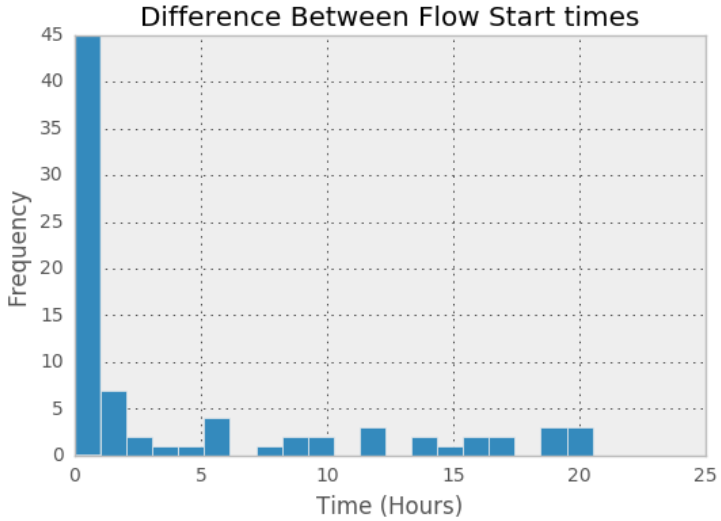

Figure 7: Frequency of the start time difference between the Windows 7 Update flows.

Figure 8 was created to see if there exist any connection between the start time difference of the flows and their sizes. It shows that the flows ranging between 667 to 5 773 bytes had a varying time difference. The flows with sizes between 181 910 - 108 654 248 bytes had an average start time difference of eleven Seconds. This is an indication that the first flows in the flow bursts are between 667 to 5 773 bytes

big and all the large flows arrive in flow bursts. The 55 280 bytes big flow that is observed to arrive after twelve hours is not downloaded in a burst.

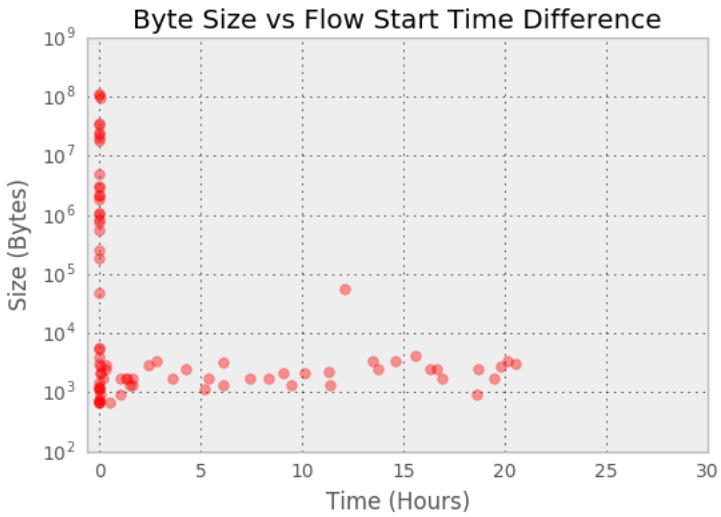

Figure 8: Windows 7 Update flow sizes versus start time difference.

5.3 Windows 8.1 Analysis

The Windows 8.1 computer was connected and searched for updates for around 24 days. During this time the computer established 7 641 flows, where 147 of the flows were classified as Windows Update. It was reveled that the computer received Windows update flows constantly during the time it was connected to the Internet with around six flows per day. The statistics for downloaded update sizes are presented in Table 8. The dataset consisted mostly of small updates around 1 300 bytes big, where only a small amount of the flows were really large.

Table 8: Size statistics for downloaded Windows 8.1 Updates.

Statistics	**Download Size (Bytes)**
Mean	279 874
Standard Deviation	1 253 812
Minimum	513
25-Percentile	926
50-Percentile	1 288
75-Percentile	1 658
Maximum	7 073 794

In Figure 9 the size of each flow is plotted against the time in days when the Windows machine downloaded the updates. The distance between each flow is the

duration from when a flow is finished to a new flow start. Out of the 147 flows two flows were in the size of 53 000 - 55 000 byte and seven flows ranged from 5 188 152 - 7 073 794 bytes received by the host. Since these flows were so much larger than the other flows in the dataset they were believed to be updates sent from Microsoft, and the other flows were considered to be update checks that checked if any updates were available. As can been seen in the figure most of the flows were downloaded individually with a couple of hours in between each other. Some small flows bursts can be observed when the larger flows are downloaded, where the duration between the flows are just a couple of seconds.

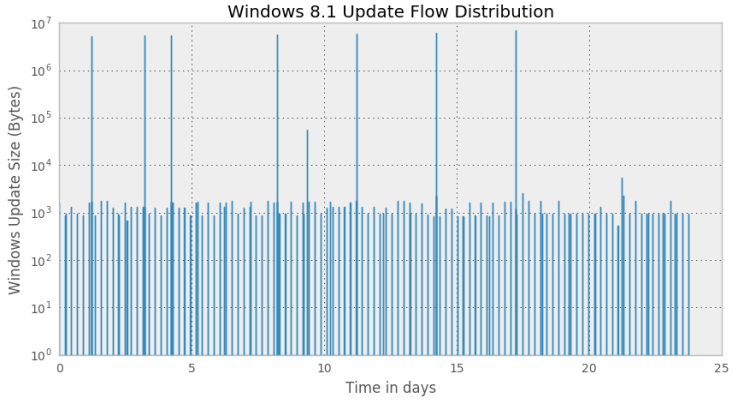

Figure 9: Distribution of Windows 8.1 Update flows with the time plotted against the amount of bytes they download.

In order to check this theory the size and the date when the updates were received was compared to Microsoft's Update Catalog [16] which contains information about all updates they have sent out and the findings is presented in Table 9. As can be observed in the table, the size and the date do not match the ones retrieved and several large updates that have been downloaded are not present in the Microsoft catalog.

Table 9: The largest Windows Update flows found in the dataset with their date, compared to the updates found in Microsoft's Update Catalog.

Windows Updates from file			**Windows Updates from catalog**	
Date	**Size**		**Date**	**Size**
01/03/2017	4.9478 MB			
03/03/2017	5.088598 MB			
04/03/2017	5.156858 MB			
08/03/2017	5.451539 MB			
11/03/2017	5.651487 MB			
14/03/2017	5.92254 MB		12/03/2017	4.8 MB
17/03/2017	6.746096 MB		16/03/2017	786 KB

When comparing the time differences between the different update flows it was revealed that the seven largest flows present in Table 9 started on average 26 seconds after a smaller flow. This is shown in Figure 9 where it it denser around the large flows. A histogram of the time difference between the start times is presented in Figure 10. The figure reveals that 77 of the 147 flows started around five hours between each other.

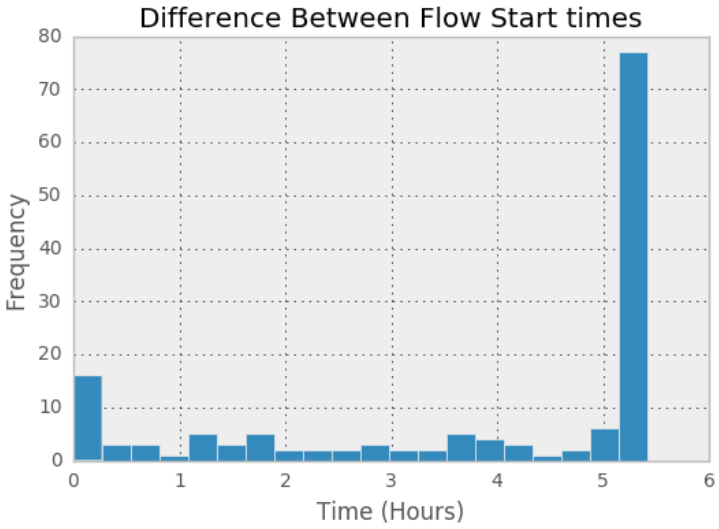

Figure 10: Frequency of start time difference between the Windows 8.1 Update flows.

In order to examine these flows more closely a new histogram was created and is presented in Figure 11, where the flows which started five hours between each other appears. It shows that the majority of the flows start after a nearly constant time have passed, in a interval of around two minutes. This indicates that the computer checks for updates regularly after a constant time have passed.

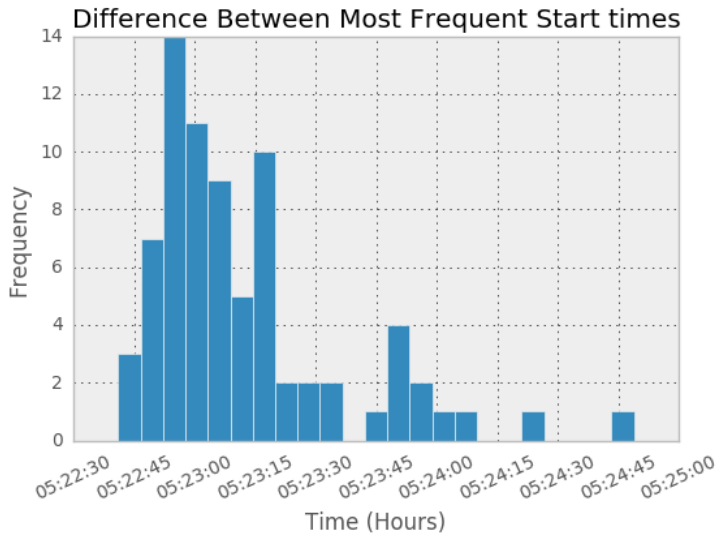

Figure 11: Zoomed in on the frequency for the 5 hour start time difference for Windows 8.1 Update flows.

In order to see if there is any difference in size between the flows with different start times, a scatter plot was created and is visible in Figure 12. The figure plots each flows size against the time difference between the start of each flow. As stated previously the majority of the flows are around 1 300 bytes big where the majority of the flows start five hours after each other. Two clusters are visible in the figure, one cluster shows that all of the 7 flows start nearly instantly after a previous flow have started. They are 5 188 152 - 7 073 794 bytes big. The other cluster consists of all the flows which have a time difference of five hours. They are 526 - 2 537 bytes big with an average of 1 277 bytes. This could be an indication that an update check have been performed and an available update have been found. This starts the download of the larger update directly after the smaller update.

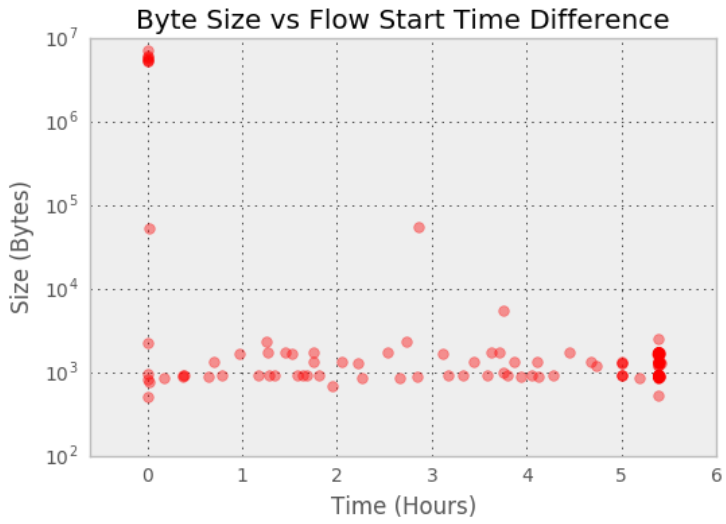

Figure 12: Windows 8.1 Update flow sizes versus their start time difference.

5.4 Windows 10 Analysis

The Windows 10 dataset consisted of 20 058 flows gathered during the same time as the Windows 7 which was approximately 15 days; out of these flows 267 flows were classified as Windows Update flows. The statistics for the downloaded Windows 10 updates are presented in Table 10. As can be seen in the table most of the flows were quite large in comparison to the flow size from the other datasets and the majority of them were one MB big.

Table 10: Size statistics for downloaded Windows 10 Updates.

Statistics	**Download Size (Bytes)**
Mean	2 533 946
Standard Deviation	3 571 159
Minimum	651
25-Percentile	1 088 050
50-Percentile	1 088 317
75-Percentile	1 092 812
Maximum	16 324 660

The size of each flow and their start time is displayed in Figure 13, here the distribution of the flows and their sizes are clearly presented. As can be observed nearly all of the flows have the same size and they arrive in 15 flow burst, where the burst starts after a close to constant time has passed. The amount of flows in the bursts varied from three up to 87 flows, with an average of 17.8 flows per burst which was also the average amount of Windows Update flows per day.

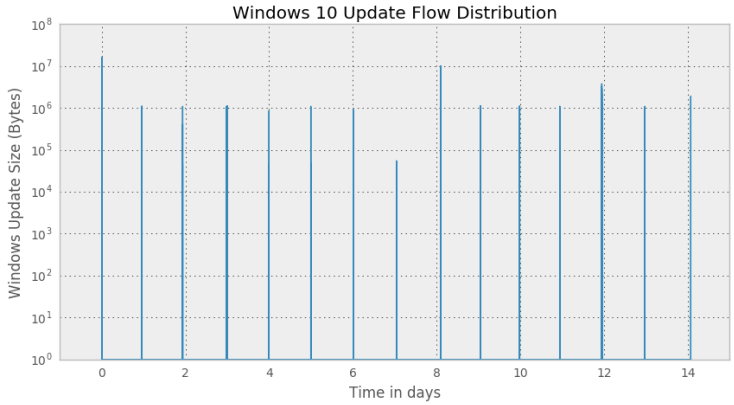

Figure 13: Distribution of Windows 10 Update flows with the start time against the amount of bytes they transfer.

In Figure 14 the frequency of start time differences is presented in order to get a better presentation on how the flows are correlated. It reveals that nearly all of the flows start instantly after each other and that 14 of the flow burst shown in figure 13 start 22 - 26 hours after each other, with an average of 24 hours between them. These are the first flows in the flow bursts.

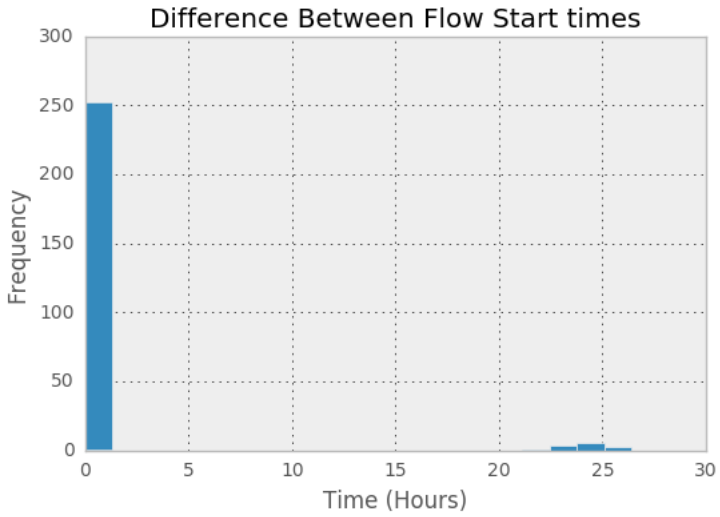

Figure 14: Frequency of start time difference between the Windows 10 Update flows.

Figure 15 reveals that nearly all of the flows which have a high time difference is 39 369 - 168 412 bytes big with an average of 56 729 bytes. One outlier is observed that is 1 911 244 bytes big which is much larger than the other flows which have a time difference of 22 - 26 hours.

The rest of the flows which start nearly instantly after each other have a varying size ranging from 651 - 16 324 655 bytes, with an average size of 2 674 141 bytes.

This reveals that the first flows in a flow burst are 39 369 - 168 412 bytes big and that nearly all of the large flows arrive in bursts.

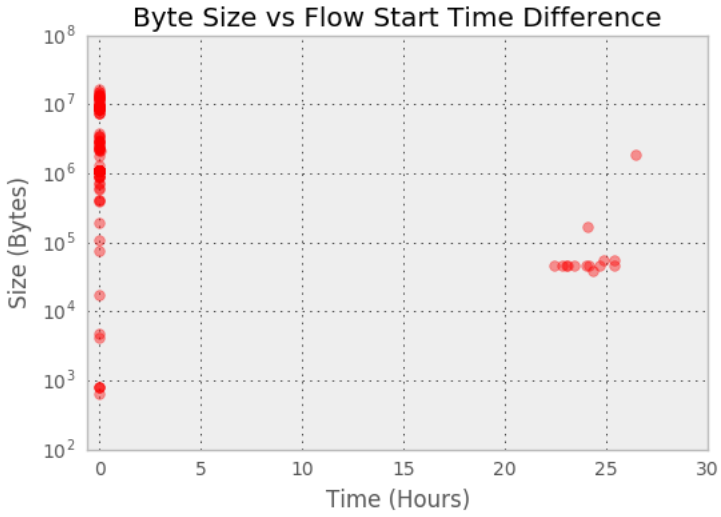

Figure 15: Windows 10 Update flow sizes versus start time difference.

5.5 Chapter Summary

The distribution for the Windows Update flows differ a lot between the different OSes, but some common characteristics can be observed. For Windows 7 and 10 nearly all of the Windows update flows were downloaded in bursts, where the time difference between each flow burst ranged from a couple of hours to nearly a day. The Windows 8.1 OS differed quite a lot in comparison to the other OSes, where nearly all of the flows arrived individually with 5:22:30 - 5:24:45 hours between each flow. All of the larger flows did arrive in flow bursts, where a smaller flow was received first and then followed by a large flow.

On average Windows 7 and 8.1 received 5 - 6 flows per day and Windows 10 received 17.8 flows per day.

In the Figures 8, 12 and 15 which show the sizes of each flow versus the start time difference between each flow reveals some similarities between the figures. The flows which have a large time difference are most often 1 - 100 KB big and the flows in the 1 - 100 MB size start directly or just a few seconds after another flow.

The detection model will be based on the attributes of each different OS and the similarities between them, but in order to come to more accurate conclusions regarding the behavior of the Windows Update flow traffic, further research have to be performed for the different OSes. Data collection for longer periods would be an improvement in order to see if the patterns observed are the same for longer durations. It can also be interesting to collect traffic from several computers with the same Windows OS to see if they all share the same pattern.

6 Design of Detection Methods

This section explains the different methods that are created and used for this study to determine if an IP-address consists of a NAT and to count the number of hosts. Chapter 3 focused on related studies and explained different methods used for NAT detection and host counting. These methods are going to analyze the application flows that a host receives in order to detect NAT and to determine the number of hosts in the NAT network. A Deep packet inspection (DPI) engine gathers the packets, identifies flows from an ISP and places them into a large dataset that will be used for the analyses. Features from the dataset are collected in order to perform NAT host detection.

The distinct amount of hosts in the NAT network is not going to be explored in detail, the most important part is to be able to detect huge networks from the datasets. The information can be used by ISPs to detect if their customers are sharing their Internet access in an unlawful manner.

6.1 NAT Detection Method

This will be the first method and it will be used to select the IP-addresses which could be identified as NAT routers in the dataset.

As mentioned in Section 2 and 3 a NAT may consist of several different devices where each device generates its own network traffic. In Section 3 some detection models were based on the amount of traffic the host could generate. Some of the most important features in this detection model were the number of packets sent and retrieved, the number of bytes exchanged, the time of a flow etc. Therefore the first step will be to identify the IP-addresses which contain the highest number of flows, the largest amount of bytes exchanged and the flows which are active for the longest durations.

When the IP-addresses which have generated the most traffic have been identified they have to be analyzed in order to determine if the traffic was generated by one or several hosts. The analysis will be performed by identifying which types of flows that the traffic consisted of. If an IP-address generates a lot of traffic as well as consisting of a large amount of NAT detection flows it is highly likely that there are several hosts using a single IP-address.

6.2 NAT Host Counting Methods

In Section 4.4 the NAT detection aspects mentioned in Table 5 and 6 are introduced and they will be used to identify different hosts for each unique IP-address. The first factor taken into account is the amount of NAT detection flows an IP-address

receives. This factor is determined based on the behavior of a NAT network because several hosts receive more updates than a single host would. Therefore if the number of updates are unusually high for a single IP-address it could be an indication that several hosts are hidden behind a NAT. With this statement in mind, several different methods have been created that will be used to analyze the data and to find patterns which could indicate NAT host behavior.

Windows Update Analysis Methods

One of the analysis methods used in this study is to identify the different hosts by examining the Windows Update flows. This method is based on the results from the empirical study of the Windows Update flow distribution in Section 5.

By analyzing the Windows Update flows from the different Windows OSes the characteristics of the flows are be used as ground truth in order to identify individual hosts. The Windows Update distribution for the datasets will first have to be analyzed in order to see if there exist common characteristics between the datasets under examination and the results retrieved from the empirical Windows Update study.

The first analysis will be performed on the size and arrival time of each Windows Update flow in the dataset. This is done in order see how the flows are distributed and to identify similarities between the flows. With this analysis, it is possible to detect common updates that several hosts are downloading but in different time periods. If common updates are detected they can be used in the NAT host detection. This method will detect if an IP-address downloads the same update several times which can then be identified as several hosts on the IP-address which downloads the same update.

The next analysis of the Windows Update flows will be similar to the analysis done in the Windows Update empirical study in Section 5, where the frequency of time difference between the flows, the size and time difference between each flow were studied. This analysis is done to compare if the results from the empirical Windows Update study are the same as on the datasets provided by Procera Networks. If similarities exist between the datasets it is possible to use the knowledge of how flows are distributed from the Windows empirical study to determine the number of hosts.

The last Windows Update analysis will evaluate individual IP-addresses flow distribution. The size and amount of flows will be analyzed against their arrival time. With this analysis it might be possible to detect common characteristic between the Windows Update flows for each IP-address, which can be used in order to update the detection model.

It was observed that around 6 - 18 flows arrived per day for a single IP-address. By using this knowledge it is possible to make a rough estimate of how many hosts that may exist behind a single IP-address. This is done according to equation 1. Where A is the number of hosts. B is the average number of Windows Update flows per day for an IP-address. C is the average amount of flows per day for the three OS's discovered in the Windows empirical study in Section 5.

$$A = B/C \tag{1}$$

As have been observed in Section 5 on the Windows Update characteristics where most of the flows arrived in flow bursts with a couple of hours to more than a day between the bursts. By using the characteristics of these bursts they can be used to detect individual hosts in a NAT.

If flow bursts are found on the IP-addresses in the datasets the average sizes of the flow bursts from the Windows Update Empirical study can be used to determine how many hosts the IP-address may consists of. The average amount of flows in the flow bursts is presented in Table 11. The flow burst sizes can be used to draw conclusions on how many hosts that might receive Window Update flows during the same time. The equation used to determine the number of hosts by using flow burst sizes are presented in equation 2. Where D is the amount of hosts. E is the amount of Windows Update flows an IP-address receives during a short interval. The interval can vary from a couple of minutes to hours. During this interval the IP-address receives a large increase in the number of flows which usually results in a peak as can be observed on IP-address number 1 in Figure 20. F is the average amount of flows in a flow burst displayed in table 11.

Table 11: Average number of flows in a flow burst.

Windows OS	**Average flow burst sizes**
7	3.56
8.1	2
10	17.8

$$D = E/F \tag{2}$$

Antivirus Analysis Method

Another analysis method used to determine the number of hosts for a IP-address is to count the number of different antivirus flows each IP-address receives. This method is supported by the fact that a regular host would only use a single antivirus for

their computer because the usage of several different antiviruses on one computer can cause problems for the user. For example, the antiviruses might cancel each other out, use up a lot of the computer systems memory or cause other problems [27]. Therefore if an IP-address receives updates from more than 1 antivirus it is highly likely that it exists at least that many hosts on the connection.

The analysis method works by trying to identify how many different antivirus flows each IP-address receives. The model will identify all antiviruses detected for a single IP-address as well as their flow amount. The IP-addresses which are revealed to contain flows from two or more different antiviruses will be grouped together and presented to the user.

Windows and Antivirus Analysis Method

The Windows and antivirus analysis method examines Windows Update and antivirus flows for individual IP-addresses. The point with this analysis is to detect hosts by observing similarities between the time an IP-address receives Windows and Antivirus flows.

If an IP-address receives a lot of Windows Update flows during a short interval it is difficult to determine if those flows arrive from just one host or several different hosts. By observing when the IP-address receives antivirus flows as well as the Windows Update flows, the host classification can be done with increased accuracy.

An example is if an IP-address receives a lot of Windows Update flows as well as flows from two different antiviruses in the same interval. This could be an indication that this particular IP-address contains at least two hosts that receive their Windows Updates during the same period.

The method will go through the IP-addresses in the dataset which contains Windows Update and at least two different antivirus flows. These IP-addresses will then be analyzed in order to see how the flows are distributed.

6.3 Chapter Summary

This chapter have explained different methods that will be used to detect NAT and to count the amount of hosts on the datasets. Three different approaches are used when trying to determine the number of hosts: Windows Update analysis, Antivirus analysis and a combination of them. The Windows Update analysis method are derived from the results achieved in Section 5. The antivirus analysis method are based on the statement that there exists only one antivirus per host. By combining these methods there will be a high chance that the IP-addresses matching these conditions may contain a hidden NAT network.

7 Procera Networks Dataset Evaluation

This part of the study will analyze the datasets provided by Procera Networks. Three different datasets that have been gathered for different durations and in different countries will be evaluated to see if they show any behavior of a NAT. The methods used to analyze these datasets have been presented in Section 6.

7.1 Small Cellular Dataset Evaluation

Dataset Overview

The first analysis was done on a Cellular dataset from Sweden provided by Procera. This test was performed to gain a basic knowledge of how the data looked like. The data was collected during approximately three hours from a mobile device network. In Figure 16 information regarding the dataset can be found and the number of unique NAT detection flows which it consisted of. As can be seen in the figure it contained 781 793 flows and from them, 3 205 were identified as the NAT detection flows, which is around 0.41% of the flows. Not all of the NAT detection aspects listed in table 5 and 6 were present in the dataset, those found are presented in Figure 16, where Windows update received the highest amount of flows and the majority of NAT detection aspects found were antivirus updates.

Total Number of Flows: 781 793
Number of Unique IP-addresses: 11 024
Number of NAT Detection Flows: 3 205
Number of Unique IP-addresses with NAT Detection Flows: 149

NAT Detection Aspects found in the Dataset:

Aspects	Count
Windows Update	1 352
Microsoft BITS	1 082
Symantec LiveUpdate	200
NOD32 update	141
APT	136
360 AntiVirus	55
Sophos Anti-Virus	41
Apple Software Update	37
Kaspersky update	35
Trend Micro AntiVirus	34
AVG Anti-virus update	33
Java update	25
McAfee Personal Firewall	15
Bitdefender Antivirus update	11
F-Prot Antivirus update	4
Avast! antivirus update	3
Adobe Update Manager	1

Figure 16: Small Cellular dataset information.

All IP-addresses with their total amount of flows were examined to see if there were any IP-address that received higher amounts of flows than the others. For the detection it is important to look on how much traffic each IP generates since more hosts create more traffic. If an IP-address has more flows than the majority of IP-addresses it could be an indication that it consists of several devices. In Figure 16 it can be seen that the dataset consisted of 11 024 unique IP-addresses, the total amount of flows each of these IP-addresses have is represented on the x-axis of a Cumulative Distribution Function (CDF) in Figure 17. The IP-addresses are represented in percentage on the Y-axis. The distribution of the CDF plot shows that a little more than 60% of the IP-addresses only have one flow. Out of all the 11 024 IP-addresses 2 586 IP-addresses were revealed to contain 771 642 of flows which is 98.7% of all the flows in the dataset. All the NAT detection flows are also present in this set. This shows that only a few IP-addresses contained the majority of the traffic which is an indication of NAT behavior.

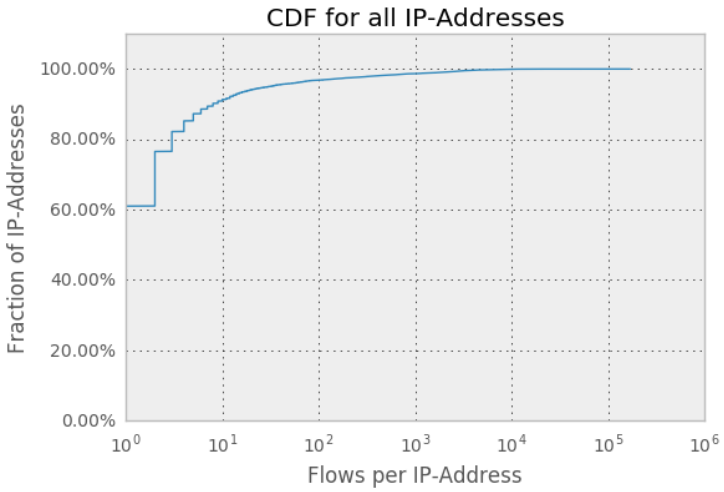

Figure 17: CDF for the logarithmic amount of flows per individual IP-address in the Small Cellular dataset.

Figure 17 revealed that some IP-addresses received a very high amount of flows compared to the rest in the set. These IP-addresses were further examined and it showed that 145 of the IP-addresses contained 674 186 flows together and 2 838 of the NAT detection flows. The left part of Figure 18 displays the 15 IP-addresses which received the highest amount of flows in the dataset. The figure shows that one IP-addresses contained 168 100 flows which are 21,5% of all the flows in the dataset. Since these IP-addresses contained so many more flows than the other ones it is a possibility that they consist of more than one host and are further examined.

In order to detect the number of different hosts in the IP-addresses, the NAT detection flows were analyzed to see if they could reveal a pattern that shows evidence for several devices. Therefore it was interesting to see how many NAT detection flows the IP-addresses with the highest amount of flows received. In the right part of Figure 18 the same IP-addresses as on the left part are displayed but only with their NAT detection flows amount. As can be observed in the figure most of the IP-addresses received a low amount of NAT detection flows where one IP-address received a lot more flows than the rest. This revealed that even if some IP-addresses contain a lot of traffic it does not have to be in proportion to the flows examined in this study. Most of the IP-addresses in Figure 18 contained a large amount of DNS, SSL and uTP flows. IP-address 167772224 contained 167 212 DNS flows.

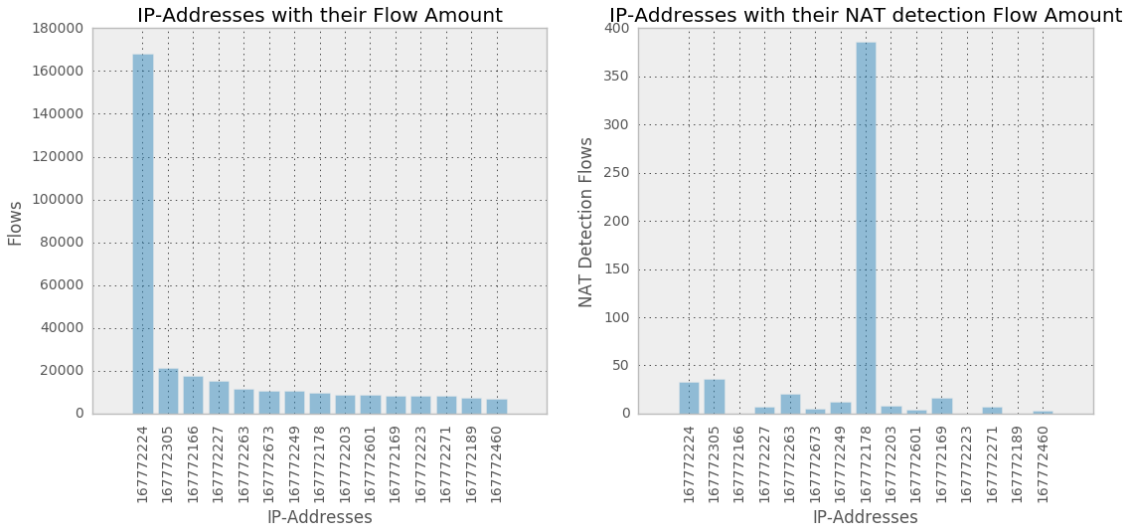

Figure 18: The 15 IP-addresses which received the highest amount of flows in the Small Cellular dataset. The left plot show the total amount of flows per IP-address and the right plot shows the same IP-addresses but with their amount of NAT detection flows.

NAT Detection Flow Evaluation

The distribution of NAT detection flows per IP-addresses in the dataset was examined and is presented in Figure 19. The left side of the figure shows that only a small fraction of the IP-addresses contained NAT detection flows. 149 of the 11 024 IP-addresses received all of the 3 205 NAT detection flows. The distribution of the NAT detection flow of the 149 IP-addresses are presented on the right part of the figure. As can be seen in the figure the majority of the IP-addresses consists of 1 - 50 flows, where some outliers received a much higher amount of NAT detection flows than the rest. Some of those IP-addresses are presented in Table 12.

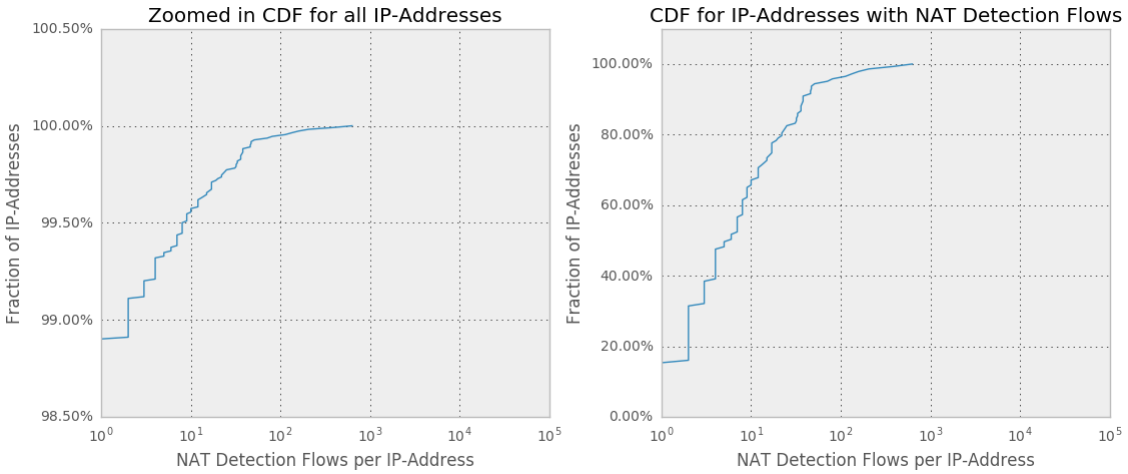

Figure 19: CDF for the logarithmic amount of NAT detection flows per individual IP-address in the Small Cellular dataset. The left plot is zoomed in on the fraction of IP-addresses which received NAT detection flows. The right plot shows the fraction of IP-addresses which only received NAT detection flows.

Since this study focuses on the evaluation of NAT host detection which uses the NAT detection flows, it is unfortunate that such a small amount of flows were discovered. The low amount of NAT detection flows may be the cause of using traffic flows from a cellular network.

Table 12: The 10 IP-addresses with the highest number of NAT detection flows in the Small Cellular dataset.

IP-address	**NAT detection flows**
167772542	629
167772178	386
167772422	206
167772179	159
167772171	133
167772209	113
167772315	81
167772478	71
167772411	51
167772181	47

Windows Update Flow Evaluation

In this section, the Windows Update flows will be analyzed in order to reveal NAT and to count the number of hosts. Of the 11 024 IP-addresses in the dataset only

84 of them contained all the Windows Update flows.

The five IP-addresses with the highest amount of Windows Update flows were then visualized against the time when the flows began to download in Figure 20.

As can be seen on the pattern of IP-address number 2, it consisted of 152 flows during a 93 minutes interval. This is a lot of flows during a short period of time if comparing to the Windows Update flows in Section 5 where the OSes received around 6 - 18 flows per day. This shows that IP-address number 2 may contain as many as 8 - 25 hosts according to equation 1.

By examining when the different flows start, it can be observed that it creates a stairway shape. This is similar to the flow burst shapes in Section 5 for the Windows 10 computer, but the time between the flows are very short. Because the flow bursts are so large and they start with just a couple of minutes in between them it could be an indication of several hosts. This can be modeled as a number of different hosts that starts their Windows Update download at different times of the day. Of the five IP-addresses examined this way all but IP-address number 3 shows this behavior. However, it is not possible to tell if several hosts receives the flows in the same interval or if it is just a single host.

35 of the IP-addresses in the dataset contained five or more Windows Update flows. The shape for nearly all of them was almost vertical where during a short period of time all the flows were received.

Some IP-addresses received flows for a short period followed by a pause and then they started to receive a lot of flows again so that it created a stairway shape as can be seen on IP-address 2, 4 and 5 in Figure 20. This shape was only observed for the IP-addresses which received a lot of flows, where none of the other IP-addresses showed this shape.

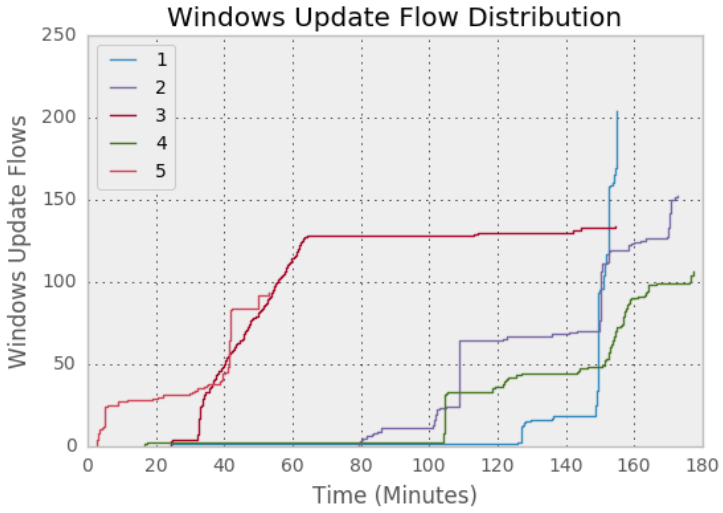

Figure 20: Windows Update flow start time versus the number of flows for different IP-addresses in the Small Cellular dataset.

Antivirus Flow Evaluation

If an IP-address contains flows from several different antiviruses, it could be an indication that it may consists of more than one host. Therefore comparison between different antivirus flows were conducted in order to gain knowledge if some of the IP-addresses received updates from several different antiviruses. The experiment revealed that five IP-addresses gained updates from two different antivirus software's, with only a small amount of updates for each software which can be seen in Table 13.

Table 13: IP-addresses which received flows from two antiviruses.

IP-address	360 AntiVirus	Symantec	AVG AntiVirus	Avast antivirus	Kaspersky	McAfee	NOD32
167772462	0	19	0	0	0	1	0
167772511	0	8	2	0	0	0	0
167772263	0	2	0	0	12	0	0
167772718	11	0	0	1	0	0	0
167773353	1	0	0	0	0	0	3

An analysis was performed on the IP-address which consisted of Windows Update and at least two antivirus flows. By observing when an IP-address receives the flows it might be possible to see if some of the flows are received during the same interval. This method can be used in order to determine if there are several hosts that receive Windows Update flows during the same interval because a host usually only have one antivirus on their computer. If there are two antivirus flows as well as a lot of Windows Update flows noticed in the same interval, it can be modeled as two hosts both received their updates during the same time.

Table 14 shows the IP-addresses which consists of both Windows Update flows and at least two different antiviruses. It was observed that the IP-addresses received their antivirus flows in the same intervals as the Windows Update flows. Furthermore, the shape of the flows were very similar which showed the same stairway shape as in Figure 20.

Table 14: The IP-addresses which contains Windows Update and two different antivirus flows.

IP-address	**Windows Update**	**360 AntiVirus**	**Kaspersky**	**NOD32**	**Symantec**
167772263	6	0	12	0	2
167773353	1	1	0	3	0

7.2 Large Cellular Dataset Evaluation

Dataset Overview

The second dataset provided by Procera was gathered from the same type of network as before but for 18.7 hours and with more activity per hour. This resulted in a much larger dataset than the first one, as can be seen in Figure 21. This dataset contained 42 488 795 flows, in comparison to the previous dataset which only contained 781 793 flows. This is an increase of 54 times in the number of flows. The amount of NAT detection flows to investigate numbered to 31 731 flows which are around 0.074% of the total amount of flows in the new dataset. This is a lower ratio of NAT detection flows when comparing to the first dataset where 0.41% of the dataset consisted of NAT detection flows. This is surprising because the flows in this dataset were created by putting together shorter packets flows than in the previous dataset. This was done in order to better detect if a device was checking for an update and to receive more flows.

While comparing the amount of NAT detection flows both datasets received it can be observed that the majority of the flows were classified as Windows Update. In this dataset, the antivirus applications were more present than in the Small cellular dataset, and a lot more flows were classified as 360 Antivirus and AVG. The larger amount of antivirus flows is possibly due to the fact that the number of packets that the flows are built of is shorter in this dataset as stated previously.

Total Number of Flows: 42 488 795
Number of Unique IP-addresses: 53 091
Number of NAT Detection Flows: 31 731
Number of Unique IP-addresses with NAT Detection Flows: 3507

NAT Detection Aspects found in the Dataset:

Aspects	Count
Windows Update	13 317
360 AntiVirus	9 597
AVG	3 880
APT	1 170
Microsoft BITS	1 147
Apple Software update	807
AVG Anti-Virus update	476
Sophos Anti-Virus update	446
Symantec LiveUpdate	284
Trend Micro AntiVirus	239
Kaspersky update	119
Java update	87
McAfee Personal Firewall	85
Avast! antivirus update	46
NOD32 update	15
Bullguard update	10
Bitdefender Antivirus update	5
Rising Antivirus	1

Figure 21: Large Cellular dataset information.

In order to detect potential NAT candidates the number of flows for each IP-address is examined. In Figure 22 all unique IP-addresses are plotted against the number of flows each of them received. The total amount of IP-addresses in the dataset are 53 091, as seen in Figure 21. Some statistics on the flows is presented in Table 15.

Table 15: Large Cellular flow statistics.

Flow Statistics	**Amount of Flows**
Count	53 091
Mean	800.3
Standard Deviation	2 115.8
Minimum	1
25-Percentile	96
50-Percentile	340
75-Percentile	923
Maximum	269 963

Figure 22 show that the highest concentration of flows per IP-addresses are in the range of 100 - 2 000 flows. The figure displays a few outliers that contains a much higher amount of flows than the rest of the IP-addresses. The outliers are those that are of most importance in this study, as mentioned previously more hosts will create more traffic, therefore it is of interest to examine why these IP-addresses contained many flows.

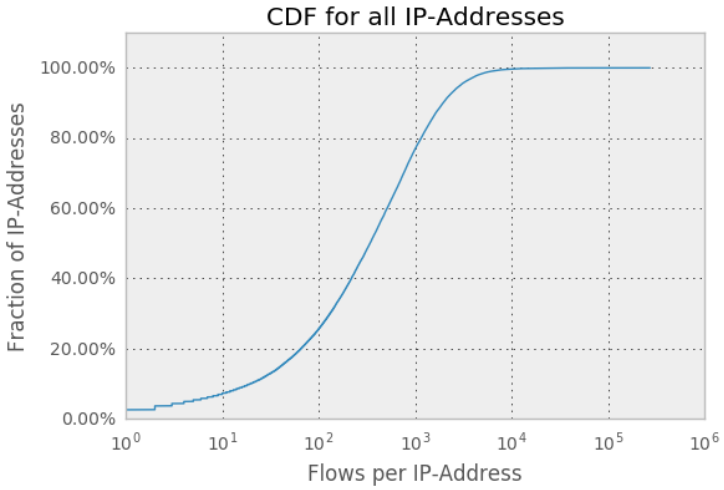

Figure 22: CDF for the logarithmic amount of flows per individual IP-address in the Large Cellular dataset.

In the dataset 5 072 IP-addresses contained 20 437 172 flows which are 48.1% of the total amount of flows. The IP-addresses which received the highest amount of flows are presented in Table 16 and are plotted in Figure 23. This shows that the number of flows some IP-addresses receive is not evenly distributed which could indicate that they may contain more than one host. As can be seen in Table 15 one IP-address contained a particularly high amount of the flows. It received 269 963 flows which are around 0,64% of all the flows in the dataset.

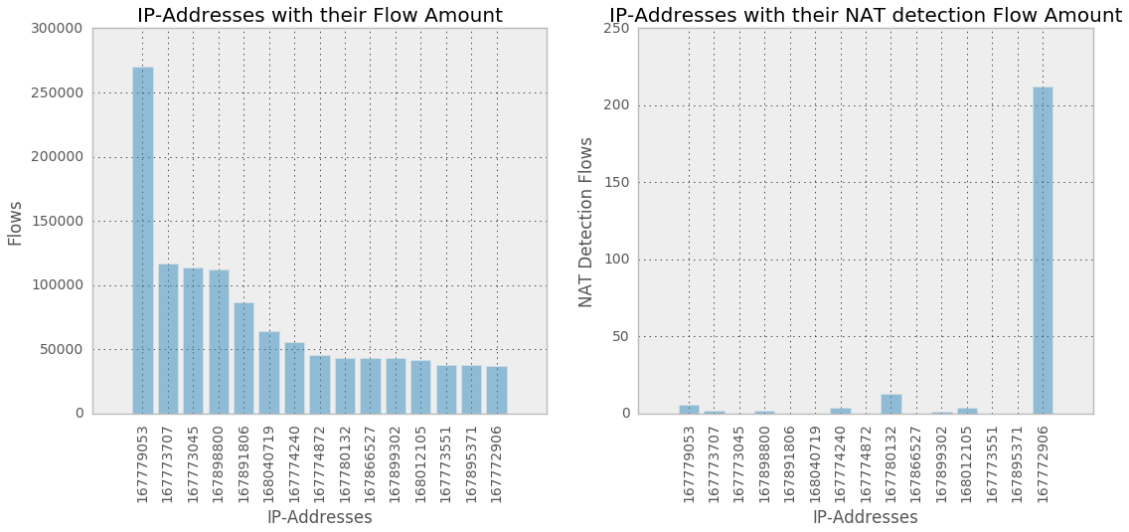

Figure 23: The 15 IP-addresses which received the highest amount of flows in the Large Cellular dataset. The left plot show the total amount of flows per IP-address and the right plot shows the same IP-addresses but with their amount of NAT detection flows.

The IP-addresses which contained the highest amount flows were further investigated in order to discover if they contained several devices. A measurement was done to discover how many NAT detection flows they contained, to see if it was in proportion to the total number of flows they received. In Table 16 and on the right side of Figure 23 the top 15 IP-addresses are displayed but only with the number of NAT detection flows they received. It was revealed that several of them did only contain a few NAT detection flows and that it was not in proportion to the total amount of flows they received. While examining what types of flows they received it was revealed that most of them contained a lot of BitTorrent flows which explains why they received such a large amount of flows.

Table 16: The 15 IP-addresses which contained the highest number of flows and with their number of NAT detection flows in the Large Cellular dataset.

IP-address	**NAT detection flows**	**Total number of flows**
167779053	6	269 963
167773707	2	116 800
167773045	0	113 501
167898800	2	112 468
167891806	0	86 670
168040719	0	64 489
167774240	4	55 541
167774872	0	45 784
167780132	13	43 272
167866527	0	43 257
167899302	1	43 126
168012105	4	42 000
167773551	0	37 874
167895371	0	37 655
167772906	212	37 362

NAT Detection Flow Evaluation

The next point of interest was to determine which IP-addresses that contained NAT detection flows and use them to detect NAT hosts. As mentioned previously, Figure 21 revealed that the dataset contained a total of 31 731 NAT detection flows, which is distributed on different IP-addresses. In Figure 24 all the 53 091 IP-addresses are represented as percentage and plotted against the number of NAT detection flows each of them contains. As can be seen on the CDF graph only a few of the IP-addresses contains NAT detection flows. Most of the IP-addresses do not contain any NAT detection flows or only a few. Out of the 53 091 IP-addresses present in the dataset, only 3 507 contained NAT detection flows, which is around 6.6% of all the IP-addresses.

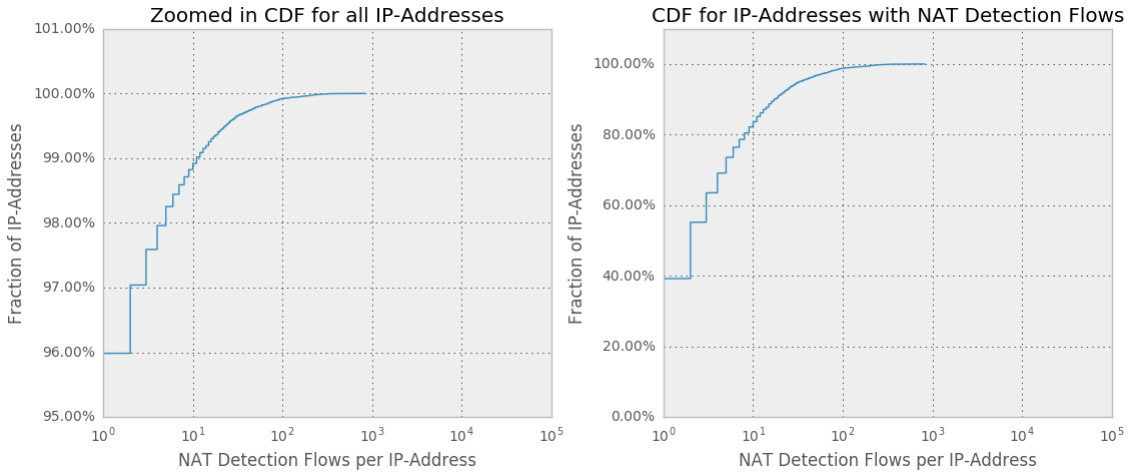

Figure 24: CDF for the logarithmic amount of NAT detection flows per individual IP-address in the Large Cellular dataset. The left plot is zoomed in on the fraction of IP-addresses which received NAT detection flows. The right plot shows the fraction of IP-addresses which only received NAT detection flows.

Analyses were performed on the 3 507 IP-addresses which contained the NAT detection flows in order to better understand how they were distributed. On the right side of Figure 24 the IP-addresses were plotted against the number of NAT detection flows each IP contained. This shows that 80% of those IP-addresses contained 1 - 10 NAT detection flows each and that some IP-addresses received over 100 flows each. By comparing this plot with Figure 19 from the small dataset, one can see that the graphs have a similar distribution. When trying to determine the amount of host for a NAT device this method examines the pattern of a certain NAT detection flow and the amount of different antivirus flows a single IP-address receives. Therefore the IP-addresses which only contained a single NAT detection flow are not examined.

In Table 17 the IP-addresses which contained the highest amount of NAT detection flows is displayed and since they received much more flows than the majority of the IP-addresses they were further examined. Most of the IP-addresses in Table 17 contained a large amount of flows from a single application, eight of them consisted mostly of Windows Update flows and two IP-addresses contained only flows from two different antivirus applications. Nearly all of the IP-addresses that received Windows Update flows also received a few Microsoft BITS flows.

Table 17: The top 10 IP-addresses which contained the greatest number of NAT detection flows in the Large Cellular dataset.

IP-address	**Flows**
167798233	821
167777945	662
167921664	347
167979988	319
167987877	315
167775183	306
167778466	276
167990933	261
168188912	254
167778788	241

Windows Update Flow Evaluation

In this part of the paper, the evaluation of the Windows Update flows will be examined. The results from the evaluation of the different OSes in Section 5 will be used as ground truth in the detection.

It was discovered that out of the total 53 091 IP-address in the dataset, 669 IP-address received all of the Windows Update flows. If each of the IP-addresses received an equal amount of Windows Update flows they would consist of one to three hosts, by comparing with the results achieved in Section 5 and equation 1.

Figure 25 displays the distribution of the downloaded and uploaded Windows Update flows with their size and start time. To gain a good representation of the flow distribution only a third of all Windows Update flows are presented in the figure. Three horizontal lines can be observed in the Downloaded Bytes graph, around the 1 KB, 10 KB and 1 MB levels. In the Uploaded Bytes section, two horizontal lines are also observed in the 1 KB and the 50 KB section. These horizontal graphs show that several flows with roughly the same sizes are found during the whole time that the flows were collected. The behavior of the flows observed in Figure 25 could indicate that Windows released a new update which several different hosts are retrieving.

Another observation in the figure is that there is a high concentration of flows in the 250 - 600 minutes interval. During these 350 minutes, 5 528 flows are received to 348 IP-addresses. That is around half of all the Windows Update flows and IP-addresses. This indicates an increase in the traffic during this period where a large proportion of the IP-addresses received Windows flows during this interval.

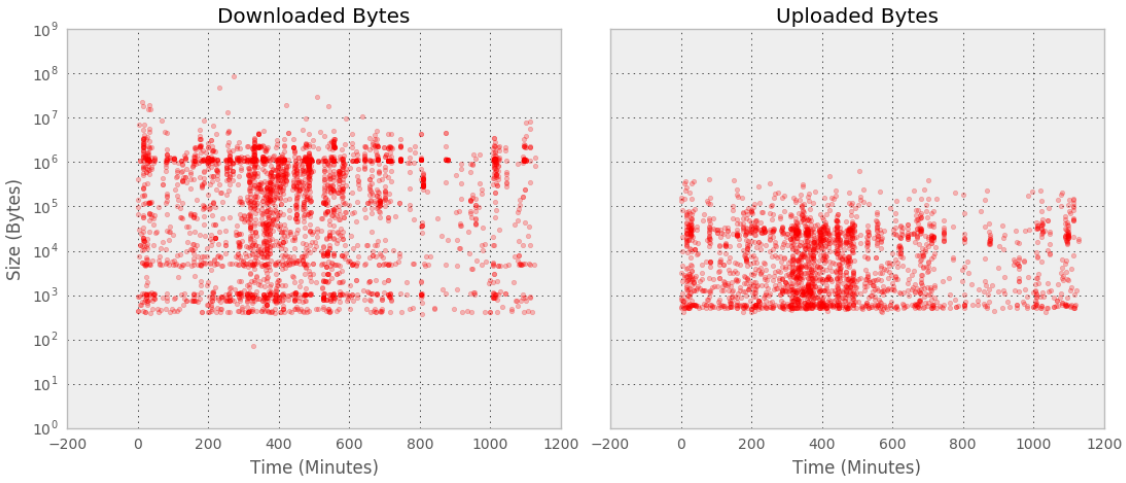

Figure 25: A third of all Windows Update flows present in the Large Cellular dataset with each flows size versus the flow start time.

Figure 26 zooms in on the horizontal line observed in the 1 MB interval in Figure 25 with all the flows present. Only a few flows with the size less than 1 090 000 bytes is observed in the figure and the difference in size between the flows are rather small. This observation further adds to the conclusion that the flows with this size are the same Windows update gathered by several hosts in different times.

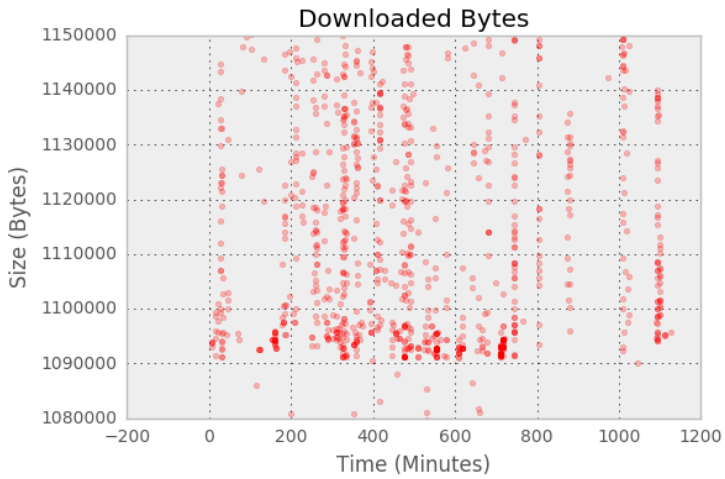

Figure 26: Zoomed in on all Windows Update flows with their flow size and start time.

Table 18 presents the statistics for the downloaded sizes of the Windows update flows. Some of the flows labeled as Windows Update flows did not contain any downloaded bytes and was therefore removed, which results in 9 650 Windows flows instead.

Table 18: Downloaded Windows Update flow sizes for the Large Cellular dataset.

Statistics	**Amount of Flows**
Count	9 650
Mean	655 346
Standard Deviation	1 730 983
Minimum	60
25-Percentile	5 501
50-Percentile	159 102
75-Percentile	1 093 103
Maximum	88 009 970

In order to see how the Windows Update flows are distributed for different IP-addresses Figure 27 was made. The figure displays the five IP-addresses which had the highest amount of Windows Update flows, where each color represent a different IP-address. The size of each downloaded Windows Update flow is presented on the left side, the number of flows on the right side and the start time of the flows are shown on the x-axis on both sides.

As can be observed in the figure all of the IP-addresses received a lot of flows during short intervals and of different sizes. There are also breaks observed for the IP-addresses where several flows are retrieved during a short interval followed by a pause and then a new burst of flows with different sizes. This behavior was mostly found on the IP-addresses which retrieved a lot of flows. The IP-addresses which received smaller amounts of flows retrieved nearly all of their flows in the same interval, with flows of different sizes.

The IP-address in Figure 27 was compared to the results in section 5 and it showed strong indications that these IP-address may contain NAT and several different hosts. The distribution of the flows were quite similar to the ones from Windows 7 and 10, where flows arrived in bursts of different sizes. However, the amount of flows the IP-addresses received was much higher than previously observed. The length of the flow bursts were also much longer. In section 5 the highest number of flows in a flow burst was observed to be 87 and the bursts only lasted for a couple of seconds. For this dataset, the bursts contained several hundred flows and could last for hours. By comparing the average flow burst sizes from Table 11 it shows that IP-address 1 might consist of 36 - 183.7 hosts depending on what OS that is being used according to equation 2. The other IP-addresses in the figure consisted of flow burst that was around 200-300 flows big and therefore may consist of around 14 - 70 hosts.

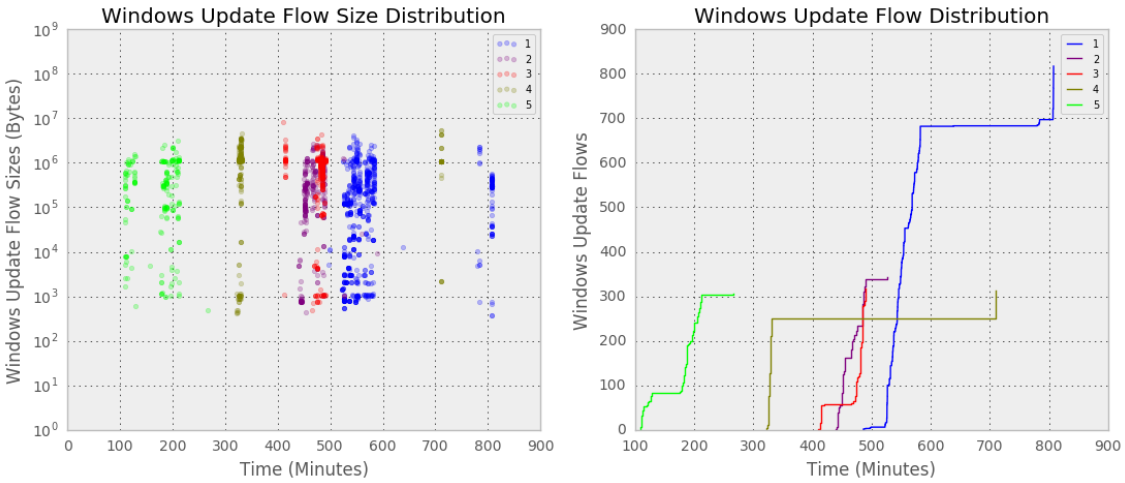

Figure 27: Windows Update flow arrival time versus the number of flows for different IP-addresses in the Large Cellular dataset.

Antivirus Flow Evaluation

Since this dataset contains several more flows than the Small dataset it might be possible to draw new conclusions of how the shape may look like when an IP-address receives flows from several different NAT detection flows. It is particularly interesting to look if there are any similarities between the Windows update and the antivirus flows start time. Therefore in order to see if some of the IP-addresses might contain several different hosts with an Windows OS the IP-addresses which had at least one Windows Update flow and two antivirus flows was compared. 21 IP-addresses contained three different NAT detection flows and one contained four different. For each different antivirus flow the IP-addresses received, it could be an indication that it have more than one host, even if the number of Windows flows are low. The IP-address which contained the highest amount of Windows Update flows are plotted in Figure 28 with all the NAT detection flows it contains.

Since there existed two services named AVG in the dataset, 'AVG' and 'AVG Anti-Virus update' they were both grouped together under the service name AVG.

In Figure 28 all the NAT detection flows IP-address 167784134 contains and their size are plotted against the time when the host initialize a connection. As can be seen in the figure the IP-address consists of Windows Update flows and three different antivirus flows. The Windows Update flows have the same shape as the ones in Figure 27 where a lot flows is retrieved during two bursts and the flows are of different sizes.

What can be observed from the figure is that the two antivirus flows start during nearly the same time as the Windows Update flows. Flows from Window Update, Microsoft BITS and McAfee are observed during nearly the same time, which could

be an indication that a computer has just been started. The host then continues to receive McAfee flows during the whole time. Furthermore, the IP-address start to receive Avast flows during the same time as the second Windows Update flow burst starts. These observations show indications that another host on the connection might start to receive their updates.

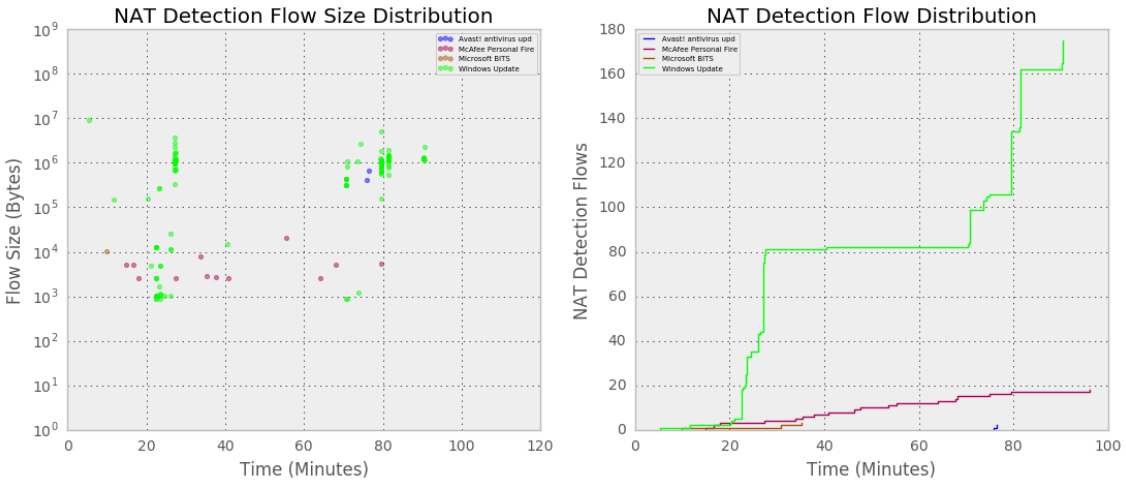

Figure 28: NAT detection flow distribution for IP-address 167784134, with the size and amount of flows versus time.

In total 66 IP-addresses were found that contained flows from at least two different antiviruses, and one IP-address were found that contained flows from three different. 42 of the 66 IP-addresses contained flows from both 360 Antivirus and AVG. On average each of the 42 IP-addresses contained 6.5 flows from 360 Antivirus and six flows from AVG.

7.3 DSL and Cellular Dataset Evaluation

Dataset Overview

The third dataset contained a mix of DSL and Cellular data which is believed to increase the amount of NAT detection flows to examine. It was provided by Procera and the traffic was collected during approximately 7 days. Figure 29 shows basic characteristics about the dataset. The dataset consisted of 68 596 470 flows in total which is 1.6 times the size of the Large Cellular dataset. What is surprising is that the number of unique IP-addresses which contained NAT detection flows is only 372, which is very small when compared to the other sets. The amount of NAT detection flows in this dataset is around 0.2% which is more flows in total when comparing to the Large Cellular dataset where 0.066% of the flows were NAT detection flows. This is interesting because that means that 372 of the IP-addresses in the dataset

shares all the 137 375 NAT detection flows, which is a lot more flows per IP-address then previously observed.

Total Number of Flows: 68 596 470
Number of Unique IP-addresses: 194 989
Number of NAT Detection Flows: 137 375
Number of Unique IP-addresses with NAT Detection Flows: 372

NAT Detection Aspects found in the Dataset:

Aspects	Count
Windows Update	97 333
AVG	13 031
360 AntiVirus	9 033
AVG Anti-Virus update	7 084
Apple Software update	3 840
Microsoft BITS	3 722
Sophos Anti-Virus update	752
Kaspersky update	731
McAfee Personal Firewall	549
Java update	429
Trend Micro AntiVirus	321
Symantec LiveUpdate	212
APT	148
Avast! antivirus update	99
Adobe Update Manager	55
NOD32 update	20
Bitdefender Antivirus update	11
Microsoft Auto Update	5

Figure 29: DSL and Cellular dataset information.

In this dataset 97 333 Windows Update flows were detected which is around the same amount of Windows Update flows as in the previous two datasets if they had been gathered for seven days. The Number of AVG flows detected was greater in this dataset than in the other ones. The NAT detection applications found in the dataset was roughly the same as in the previous ones, where most of the applications were identified as antiviruses.

Table 19 shows the statistics for the number of flows each IP-address contains. Most of the IP-addresses in this dataset did only receive a few flows, which can also be seen in Figure 30.

Table 19: DSL and Cellular flow statistics.

Flow Statistics	**Amount of Flows**
Count	194 989
Mean	352
Standard Deviation	8 042
Minimum	1
25-Percentile	1
50-Percentile	1
75-Percentile	84
Maximum	2 004 558

Figure 30 shows a CDF plot of how many flows each unique IP-address received. Nearly 60% of all the IP-addresses received only one flow which is very few since this traffic was collected for seven days. In contrast, the other 40% of the IP-addresses contained a lot of flows, where all of them contained 50 flows or more. As can be seen in the figure a small amount of the flows received significantly more flows than the other IP-addresses, where one IP-address received as many as 2 004 588 flows.

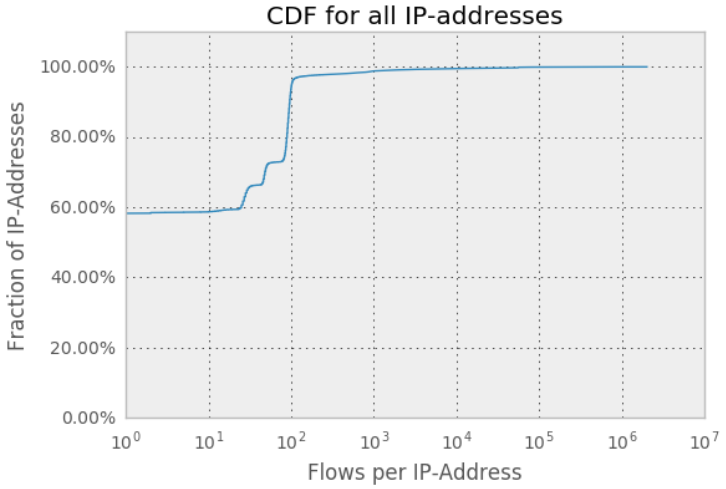

Figure 30: CDF for the logarithmic amount of flows per individual IP-address in the DSL and Cellular dataset.

It was discovered that 1 670 of the IP-address contained 60 765 572 flows together which is 88.6% of all the flows in the dataset. 136 915 of the NAT detections flows were discovered to be contained in these 1 670 IP-addresses. This is further evidence that only a few of the IP-addresses created the majority of the traffic.

In Table 20 the total amount of flows, as well as the number of NAT detection flows, is presented for the IP-addresses which received the highest amount of flows.

The IP-addresses are also represented in Figure 31. As can be observed from the figure some similarities exist between the three datasets where one IP-address often contains several more flows than the others, but for this dataset, the total amount of flows the IP-addresses received were significantly greater.

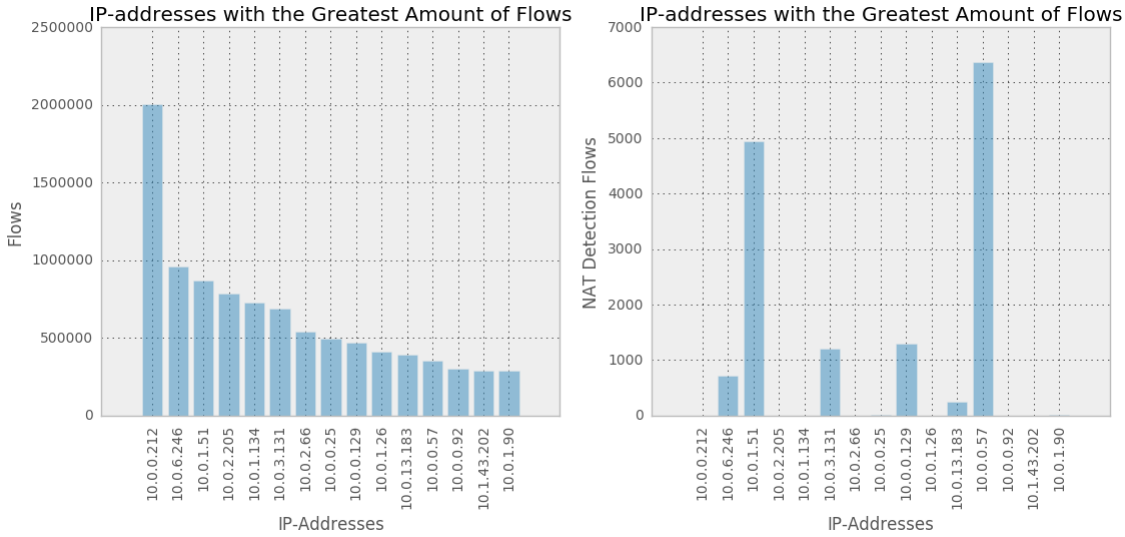

Figure 31: The 15 IP-addresses which received the highest amount of flows in the mixed DSL and Cellular dataset. The left plot shows the total amount of flows per IP-address and the right plot shows the same IP-address but with their amount of NAT detection flows.

The IP-addresses did however not contain that many NAT detection flows, where IP 10.0.0.212 did not contain a single NAT detection flow. When further examining the flows it was revealed that IP-address 10.0.0.212 did only contain Internet Control Message Protocol (ICMP) flows and most of the other IP-addresses contained a high amount of Bittorrent, ICMP and DNS flows.

Table 20: The top 15 IP-addresses with their number of NAT detection flows and total number of flows in the DSL and Cellular dataset.

IP-address	**NAT detection flows**	**Total number of flows**
10.0.0.212	0	2 004 558
10.0.6.246	710	961 375
10.0.1.51	4947	871 070
10.0.2.205	0	786 837
10.0.1.134	0	725 961
10.0.3.131	1208	690 163
10.0.2.66	0	538 657
10.0.0.25	3	494 161
10.0.0.129	1288	470 330
10.0.1.26	0	409 063
10.0.13.183	244	390 067
10.0.0.57	6370	355 045
10.0.0.92	0	299 998
10.1.43.202	0	285 949
10.0.1.90	5	285 858

NAT Detection Flow Evaluation

The distribution of the NAT detection flows for this dataset is evaluated in this section. In Figure 32 the IP-addresses which contain NAT detection flows are visible. The left plot shows how many of the IP-addresses in the dataset which contains NAT detection flows, as can be seen, it is around 0.2% of the 194 989 IP-addresses. The right side displays the 372 IP-addresses which contain all the NAT detection flows, as can be observed these IP-addresses contained a much higher amount of NAT detection flows when comparing to the two other datasets, where 60 - 80% of the IP-address only contained 1 - 10 flows each. In this dataset, the majority of the flows contained 10 - 1000 flows each which is a large increase in the number of flows each IP received, which could be an indication that there are several hosts behind one IP-address.

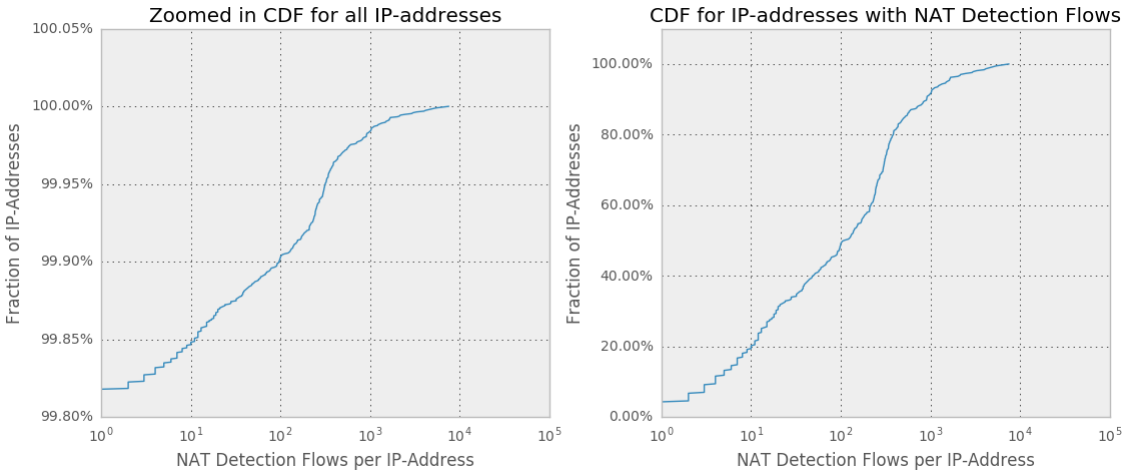

Figure 32: CDF for the logarithmic amount of NAT detection flows per individual IP-address in the DSL and Cellular dataset. The left plot is zoomed in on the fraction of IP-addresses which received NAT detection flows. The right plot shows the fraction of IP-addresses which only received NAT detection flows.

The NAT detection flows in the dataset were further examined to determine if some of the IP-addresses might contain several hosts. The IP-addresses which contained the highest amount of NAT detection flows is presented in Table 21. In comparison to the other datasets the IP-addresses of this dataset received a much higher amount of NAT detection flows where the previously highest amount of flows per IP-addresses was 821. Most of the IP-addresses in Table 21 received the majority of their flows from one application, where Windows Update, AVG, and 360 Antivirus were most present.

Table 21: The top 10 IP-addresses which contained the greatest amount of NAT detection flows in the DSL and Cellular dataset.

IP-address	**Number of Flows**
10.2.47.31	7 494
10.0.0.57	6 370
10.1.71.53	5 422
10.0.1.51	4 947
10.0.0.253	4 536
10.0.4.179	4 157
10.0.10.218	3 988
10.0.23.29	3 288
10.0.2.135	3 005
10.0.2.134	2 912

Windows Update Flow Evaluation

The first NAT detection method was performed by analyzing the Windows Update flows and the IP-addresses which received them in order to detect the number of hosts behind each IP-address. Table 22 shows the size statistics for the Windows Update flows. As can be observed the size of the flows varies quite a lot. Some of the downloaded flows did not have any size and were therefore removed.

Table 22: Downloaded Windows Update flow size statistics for the DSL and Cellular dataset.

Flow statistics	**Download size (Bytes)**
Count	96 709
Mean	1 418 866
Standard Deviation	23 772 350
Minimum	66
25-Percentile	10 812
50-Percentile	138 106
75-Percentile	1 093 127
Maximum	2 246 904 370

Figure 33 displays the distribution of downloaded and uploaded bytes for the Windows Update flows. To gain a good representation of the flow distribution the tenth of all Windows Update flows is presented in the figure. As can be seen in the Figure it is similar to Figure 25 in the Large Cellular dataset, where the same three horizontal lines can be observed in the Downloaded bytes graph. It exists some differences between the Uploaded Bytes graphs but a horizontal line can be observed in the 1 KB intervals which are similar to the previous results. This reveals that the distribution of the Windows Update flows are quite similar between the different datasets, even if the flows are gathered during several days. These results are however different from the ones determined from the Windows 8.1 experiment, which displayed only a few flows in the 1 MB sizes in contrast to this dataset. In the dataset 256 of the IP-addresses contained the 96 709 downloaded Windows Update flows, which on average leads to around 54 Windows Update flows for each IP-address per day. That is roughly around three to nine hosts per IP-address according to equation 1 which make it highly likely that several of the IP-addresses are behind a NAT.

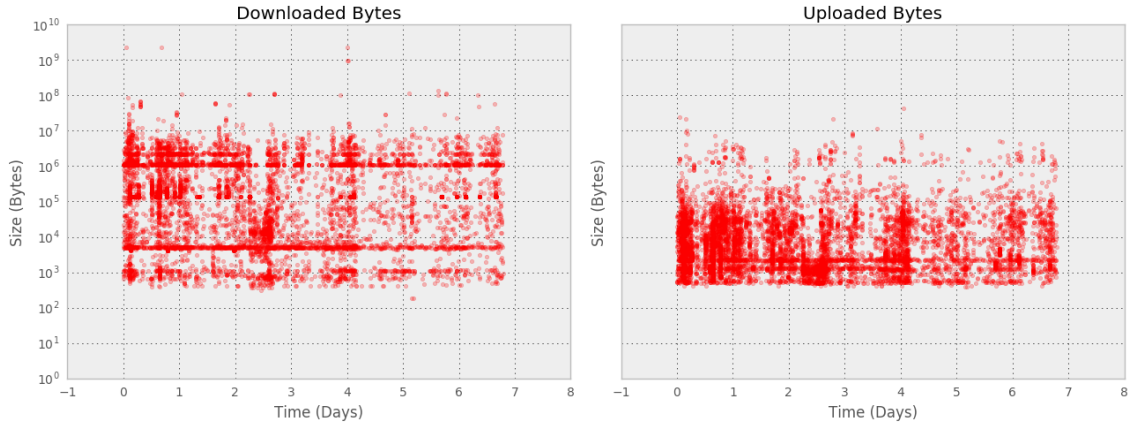

Figure 33: A tenth of all Windows Update flows present in the DSL and Cellular dataset with each flows size versus the flow start time.

Since the traffic was collected approximately for a week experiments were, performed in order to see if it existed some similarities between the Windows Update flows from this dataset and the Windows flows from the previous evaluation in Chapter 5.

An evaluation was performed to see if the time difference between the flows for each IP-address showed a similar result as with the Windows VM's. What can be observed is that the start time differences between nearly all of the flows were very short, 77 583 of the Windows Update flows started one second or less after a previous flow. Figure 34 shows only the frequencies of flows that have a time difference of one hour or more where 4 460 of the 96 709 flows are found. 2 518 of the flows have a time difference of one to two hours and 1 209 flows had a time difference of five to 160 hours. This test was not possible on the previous datasets because the traffic was collected for a much shorter duration.

As can be observed in the right side of Figure 34 the most frequent time differences are between 5 - 30 hours, 1 101 flows are found here. In this range all the flows which were the first in a flow burst from the Windows study exists, therefore some similarities between this dataset and the one from the Windows study exists.

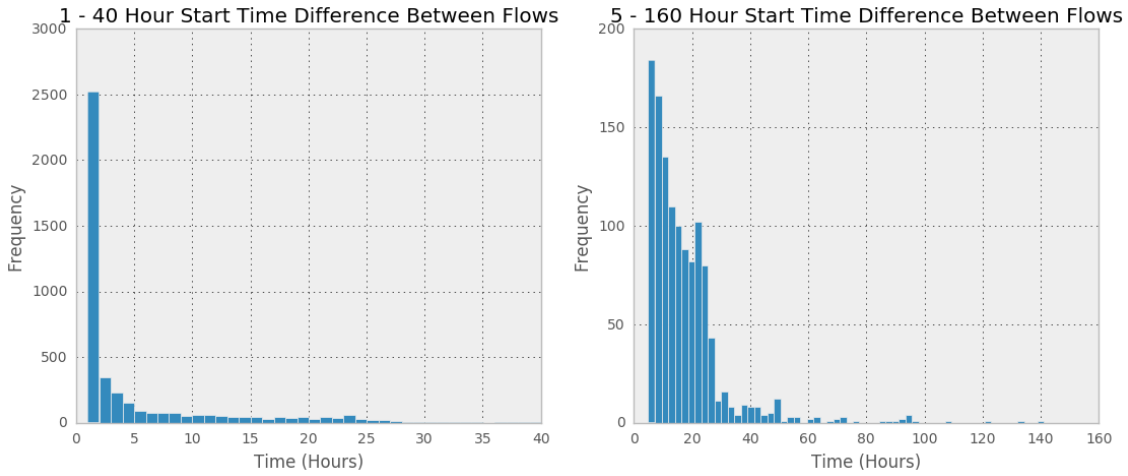

Figure 34: Zoomed in on the time difference frequency for the DSL and Cellular dataset.

The next experiment examined the distribution of the flows with their size and the time difference between them. From the analysis of the Windows OSes, it was revealed that the flows which were in the 1 - 100 MB sizes started nearly instantly after another flow and the time difference between the flows that were 1 - 100 KB could range from a couple of seconds to several hours. The longest duration between the flows was observed to be 33 hours.

As can be observed in Figure 35 the distribution of the flows is similar to the ones in the Windows OS experiments. 33 078 of the flows in the dataset were in the 1 - 100 MB size and out of those 31 915 flows were discovered to have a time difference of 0 - 20 seconds, which is similar to the results achieved previously. It can also be observed that nearly all of the flows in the 1 - 100 KB size range have a similar distribution to the ones observed in the Windows OS tests.

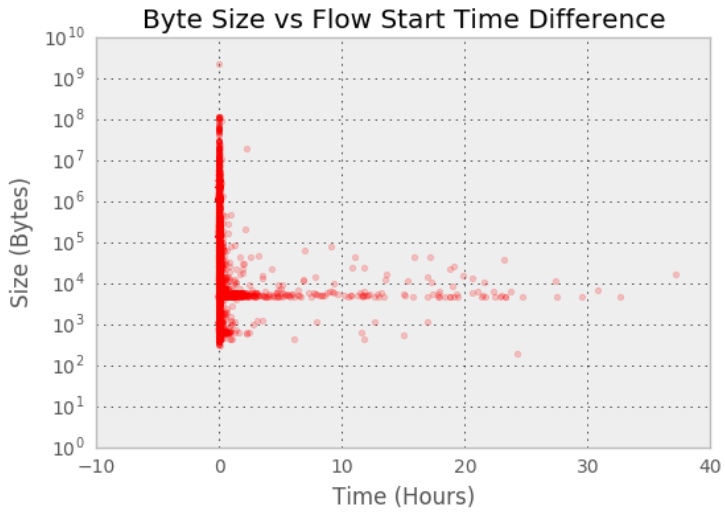

Figure 35: Windows Update flow sizes versus start time difference for the DSL and Cellular dataset. Shows every 10th flow in the dataset.

Figure 36, 37 and 38 shows all the Windows Update flows for the 15 IP-addresses which received the highest amount of Windows Update flows. The size and start time for each downloaded flow is presented on the left side. The right plot displays the number of flows each IP-address receives as well as the start time for each flow. Each color represents an different IP-address and the color is the same on the left and right side. What can be observed is that the flow distribution is rather different between the IP-addresses. Some of the IP-addresses receives several thousands of flows during just a few hours, whilst on the other hand IP-address number 1 in Figure 38 received a near constant increase of flows in the seven day period. The flows are distributed in a lot of different shapes but the shapes that were the most common are either that the majority of the flows arrive in single large burst or that they arrive over a longer period of time in smaller bursts.

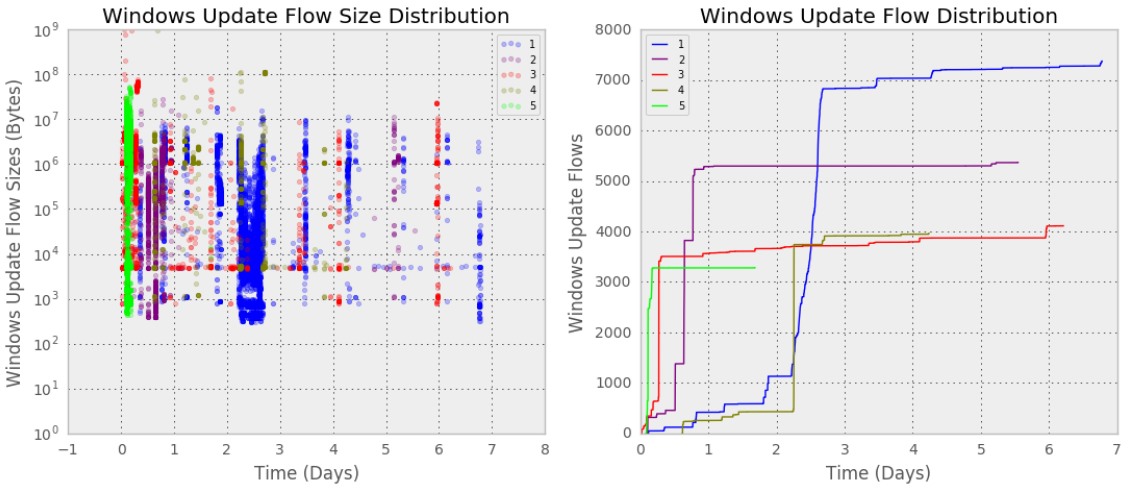

Figure 36: The size and amount of flows versus the flow arrival time for the 1 - 5 IP-addresses which contained the highest amount of Windows Update flows.

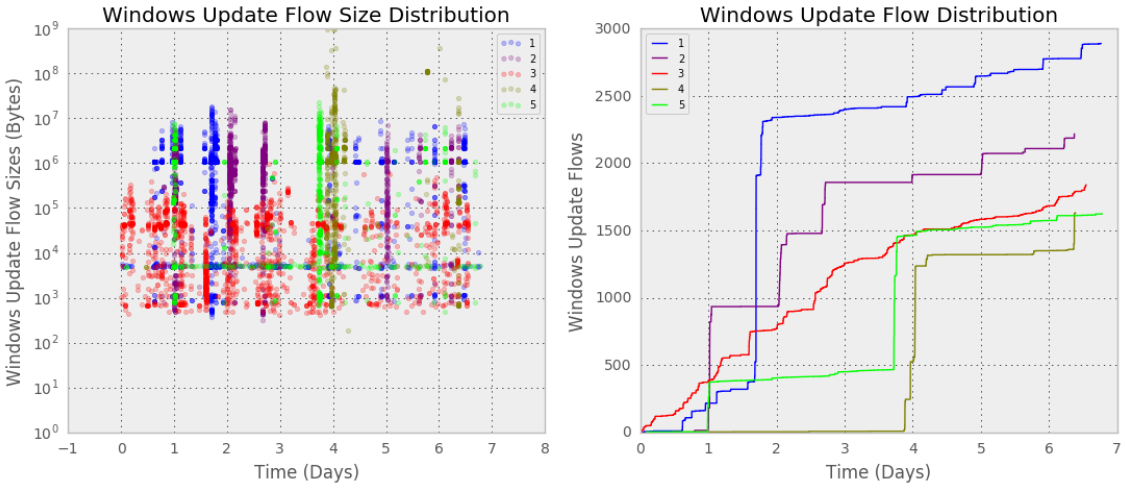

Figure 37: The size and amount of flows versus the flow arrival time for the 6 - 10 IP-addresses which contained the highest amount of Windows Update flows.

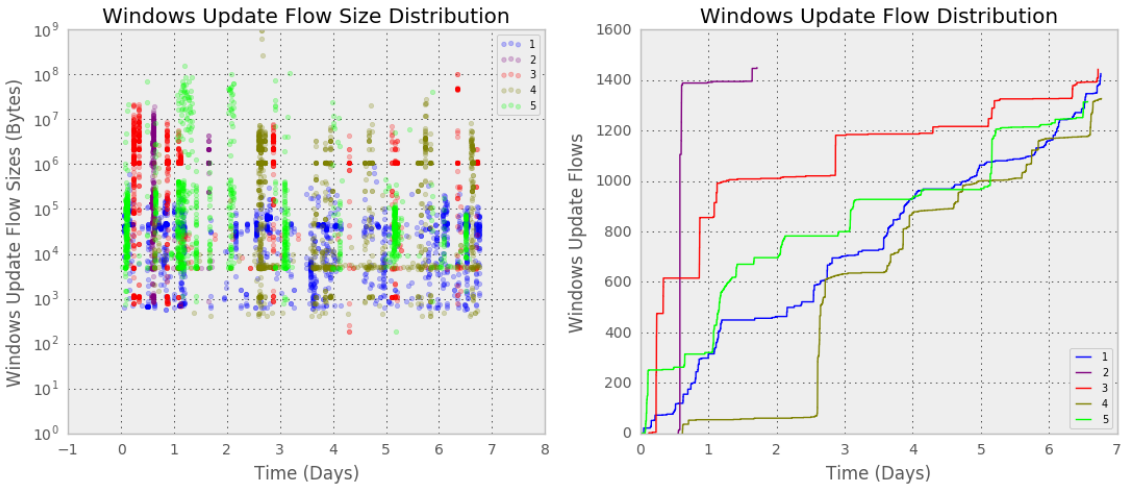

Figure 38: The size and amount of flows versus the flow arrival time for the 11 - 15 IP-addresses which contained the highest amount of Windows Update flows.

The number of flows these IP-addresses receive is several times greater than observed in Section 5. IP-address number 1 in Figure 36 for example received around 6 000 flows during a couple of hours. This is far more than previously detected. If the number of flows found during this burst was compared to the burst sizes in section 5 it would indicate that IP-address number 1 may consists of 337 - 1 685 individual hosts according to equation 2. This is an enormous increase in the potential amount of hosts for a single IP-address. However, in order to be accurate if the IP-address can consist of that many host's further research have to be made.

Antivirus Flow Evaluation

This dataset consisted of flows from both 'AVG' and 'AVG Anti-Virus', these flows were grouped together since they are from the same antivirus.

The antivirus detection tests revealed that 35 IP-addresses received flows from two or more different antivirus flows, which could indicate that these IP-addresses consists of a network with more than one host. At most six IP-addresses received flows from three different antivirus flows. The antivirus which had the highest number of flows in the dataset was AVG, and some IP-addresses received a very high amount of these flows, where one IP-address received as many as 5 347 flows.

30 IP-addresses were revealed to contain flows from Windows Update as well as two or more different antivirus flows. Where the six IP-addresses which received flows from three different antiviruses also contained Windows Update flows. It was observed that most of the flows from the antiviruses were received during the same intervals as the Windows Update flow.

7.4 Chapter Summary

The analysis of the three datasets reveals that each of them has some IP-addresses which might hold a private network with several hosts in it. However, the datasets did only contain a low amount of flows that could be used in the NAT detection. This can result in that several of the IP-addresses may contain a NAT but the methods used in this study is unable to detect them. This may be the cause of using traffic flows which are collected from cellular networks. According to the statistics from Statcounter [31] cellular networks have a low amount of Windows phones. The analysis of the antivirus flows shows there are some IP-addresses which contains flows from two or more different antiviruses. This is an indication that there might exist several hosts behind those IP-addresses. Furthermore several of the IP-addresses contained large amounts of Windows Update flows. The results obtained from the Windows Update empirical study was compared with how the Windows Updates was distributed on the three datasets. Comparison between the number of flows each IP-address received showed that several of the IP-addresses may contain a private network with different hosts in it. But in order to provide a good estimate of the number of hosts each IP-address may contain further studies of the flows have to be performed.

8 Conclusion

8.1 Summary

This study has evaluated datasets in order to discover the presence of NAT and to determine the amount of hosts behind a NAT. Methods that used flows from different applications and softwares were used for the detection.

The datasets were provided from two different sources: Karlstad university's lab and Procera Networks. The datasets were gathered with the use of a DPI engine and the packets were then grouped together into flows that were examined.

The datasets gathered in the laboratory at Karlstads University was done in order to gain ground-truth information on one of the detection aspects namely Windows Update. Three VMs were setup where each had a different Window OS. By performing tests and evaluating the properties of the Windows Update flows, attributes were found that was used in the NAT host counting analysis. It was found that the distribution of the traffic was rather different between the OSes. For some OSes, the flows arrived in large bursts with several flows arriving during the same intervals. Other OSes received a single flow that arrived after a near constant time had passed. The attributes that were discovered for each OS were then used to create new NAT detection methods that was used on the datasets provided by Procera Networks.

The datasets provided by Procera Networks were evaluated to discover if there were any IP-addresses which consisted of a NAT and the number of hosts behind the NAT. The datasets were gathered with Proceras DPI engine and three different datasets were provided for this study. The datasets were gathered for different durations and from the traffic of real cellular networks, where one of the datasets partly consisted of DSL traffic. The presence of the NAT detection flows which were used to determine the number of hosts behind each IP-address was low for all of the datasets. Around 0.41%, 0.07% and 0.2% of the flows in the datasets consisted of NAT detection flows. The reason for this low amount of NAT detection flows might be because the datasets were mostly made out of traffic from cellular networks. Since the amount of Windows phones and the usage of antiviruses on mobiles are rather low it might serve as one explanation for the low rate of NAT detection flows [31]. The low amount of NAT detection flows limits the capability of this detection model, where the majority of the flows could not be used to detect NAT or to count the number of hosts. Some indication of NAT traffic behavior were found on all the three datasets as well as some indication on how many hosts the IP-address might consist of. To provide a higher accuracy on the number of hosts behind the IP-addresses further research have to be made.

8.2 Future Work

Examine Traffic Gathered from DSL Networks

By collecting data from a real DSL network it might be possible to increase the number of flows which are classified as the NAT detection flows used in this study. This might increase the amount of Windows flows and antivirus flows. Because it is more common to find desktop computers with a Windows OS on a DSL network than a cellular network, as can be seen on the statistics from StatCounter [30], where Windows stands for around 84% of the market share for all desktop OSes. It is also highly likely that these Windows OSes have an antivirus software installed. If the amount of NAT detection flows would increase for the datasets it would provide more data to analyze and allow more accurate methods to be derived.

Labeled NAT and non-NAT Traffic

By collecting data which comes from a NAT it might increase the accuracy of the detection methods if the traffic flows were labeled as NAT or non-NAT. By knowing which flows that are NATed or not, it is possible to determine the characteristics of the flows that arrive from a NAT using that as ground truth and thereby creating more accurate detection methods. Since the flows in this study were not labeled as NAT or non-NAT it proved difficult to know if the host counting methods performed well and more experiments have to be performed to be certain of the results.

SYN Flow Detection

Another method that can be used to determine the amount of host behind a NAT is by using the SYN packet sizes. As stated by Dietz [8], the size of the SYN packet are different for each OS. By collecting the size of each SYN packet observed in the dataset, it might be possible to derive how many different OSes which sends traffic from a single IP-address. By combining this detection method with the previous ones explained in this study, it may be possible to count the number of hosts for a single connection with a higher accuracy.

Extended Windows Update Empirical Study

In order to be certain on the behavior of the Windows Update flows more analysis of the flows have to be performed. By collecting traffic from several computers with the same windows OS it is possible to make certain that the distribution of the flows is the same. It may also be possible to identify if they are receiving the same updates, which could then be used for host detection.

By gathering traffic flows for a longer duration it would also increase the accuracy of using the Windows Update flows in the detection. Because it would make the ground truth data more reliable and it would be easier to determine the maximum amount of flows a host can receive and how often they are received. By having reliable answers for how the Windows Update flow distribution look like it is possible to determine the highest number of hosts that each IP-address can consist of.

8.3 Concluding Remarks

This study has focused on analyzing datasets and deriving methods to determine the number of hosts behind a NAT. The analysis was done with Python and the Pandas package, to handle large amounts of data in the datasets and to create methods for analyzing the flows. An analysis on Windows Update flows was performed and several NAT host detection methods were derived from the results of this study. In total six datasets were examined in two different experiments. The results show that NATed hosts might exist on the provided datasets from Procera and it may be possible to determine how many hosts that exists behind each NAT with the methods derived in the study, if further research is performed. On a personal level, several new methods and knowledge have been gained during the course of the project. How a NAT works and various methods on how to determine the amount of host behind a NAT have been learned. Python programming was introduced and how to handle large datasets was also learned.

References

[1] Sebastian Abt, Christian Dietz, Harald Baier, and Slobodan Petrović. Passive remote source NAT detection using behavior statistics derived from Netflow. In *Proceedings of the 7th IFIP WG 6.6 International Conference on Autonomous Infrastructure, Management, and Security: Emerging Management Mechanisms for the Future Internet - Volume 7943*, AIMS'13, pages 148–159, Berlin, Heidelberg, 2013. Springer-Verlag.

[2] AV-Comparatives. /textIT Security Survey 2017, 2017. [Online][Cited: 2017-03-13] `https://www.av-comparatives.org/wp-content/uploads/2017/01/security_survey2017_en.pdf`.

[3] Steven M Bellovin. A technique for counting NATted hosts. In *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pages 267–272. ACM, 2002.

[4] Jun Bi, Lei Zhao, and Miao Zhang. Application presence fingerprinting for NAT-aware router. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pages 678–685. Springer, 2006.

[5] N Brownlee, C Mills, and G Ruth. RFC 2722-Traffic Flow Measurement. *Architecture*, 10, 1999. [Online][Cited: 2017-03-16] `https://tools.ietf.org/html/rfc2722`.

[6] Microsoft Corporation. Microsoft Security Intelligence Report, Volume 21. 2016. [Online][Cited: 2017-05-12]`https://www.microsoft.com/security/sir/default.aspx`.

[7] NumPy developers. Numpy, 2017. [Online][Cited: 2017-02-05] `http://www.numpy.org/`.

[8] Christian Dietz. Passive remote detection of network address translation (NAT) by using NetFlow. *Citeseer*, 2013. [Online][Cited: 2017-02-23]`https://www.dasec.h-da.de/wp-content/uploads/2013/08/thesis-dietz_Final.pdf`.

[9] Python Software Foundation. Python, 2017. [Online][Cited: 2017-02-05] `https://www.python.org/`.

[10] Yasemin Gokcen, Vahid Aghaei Foroushani, and A Nur Zincir Heywood. Can we identify NAT behavior by analyzing traffic flows? In *Security and Privacy Workshops (SPW), 2014 IEEE*, pages 132–139. IEEE, 2014.

[11] 2017 Project Jupyter. JupyterNotebook, 2017. [Online][Cited: 2017-02-05] http://jupyter.org/.

[12] Tomáš Komárek, Martin Grill, and Tomáš Pevnỳ. Passive NAT detection using HTTP access logs. In *Information Forensics and Security (WIFS), 2016 IEEE International Workshop on*, pages 1–6. IEEE, 2016.

[13] Rui Li, Hongliang Zhu, Yang Xin, Yixian Yang, and Cong Wang. Remote NAT detect algorithm based on support vector machine. In *Information Engineering and Computer Science, 2009. ICIECS 2009. International Conference on*, pages 1–4. IEEE, 2009.

[14] Gregor Maier, Fabian Schneider, and Anja Feldmann. NAT usage in residential broadband networks. In *International Conference on Passive and Active Network Measurement*, pages 32–41. Springer, 2011.

[15] Microsoft. Windows update frequency, 2008. [Online][Cited: 2017-05-12] https://technet.microsoft.com/en-us/library/cc627316.aspx.

[16] Microsoft. Microsoft Update Catalog, 2017. [Online][Cited: 2017-05-12] http://www.catalog.update.microsoft.com/Home.aspx.

[17] Microsoft. Windows Update FAQ, 2017. [Online][Cited: 2017-05-12] https://support.microsoft.com/en-us/help/12373/windows-update-faq.

[18] Sophon Mongkolluksamee, Kensuke Fukuda, and Panita Pongpaibool. Counting NATted hosts by observing TCP/IP field behaviors. In *Communications (ICC), 2012 IEEE International Conference on*, pages 1265–1270. IEEE, 2012.

[19] Procera Networks. Procera Networks, 2015. [Online][Cited: 2017-05-14] https://www.proceranetworks.com/.

[20] OPSWAT. Antivirus and Compromised Device Report: January 2015, 2015. [Online][Cited: 2017-05-12] https://www.opswat.com/resources/reports/antivirus-and-compromised-device-january-2015.

[21] The pandas community. Pandas, 2016. [Online][Cited: 2017-02-05] http://pandas.pydata.org/.

[22] Hanbyeol Park, Seung-hun Shin, Byeong-hee Roh, and Cheolho Lee. Identification of hosts behind a NAT device utilizing multiple fields of IP and TCP. In *Information and Communication Technology Convergence (ICTC), 2016 International Conference on*, pages 484–486. IEEE, 2016.

[23] Christopher Parsons. Deep Packet Inspection and Its Predecessors, 2012. [Online][Cited: 2017-05-14] https://www.christopher-parsons.com/Main/wp-content/uploads/2013/02/DPI-and-Its-Predecessors-3.5.pdf.

[24] Lisa Phifer. The trouble with NAT. *The Internet Protocol Journal*, 3(4):2–13, 2000. [Online][Cited: 2017-03-16].

[25] Jon Postel et al. RFC 791: Internet protocol. 1981. [Online][Cited: 2017-03-16].

[26] Jon Postel et al. Transmission control protocol RFC 793, 1981. [Online][Cited: 2017-03-16].

[27] Kaspersky lab Serge Malenkovich. Why using multiple antivirus programs is a bad idea, 2013. [Online][Cited: 2017-05-12] https://blog.kaspersky.com/multiple-antivirus-programs-bad-idea/2670/.

[28] Matt Smith and Ray Hunt. Network security using NAT and NAPT. In *Networks, 2002. ICON 2002. 10th IEEE International Conference on*, pages 355–360. IEEE, 2002.

[29] P Srisuresh and M Holdrege. Rfc 2663. *IP Network Address Translator (NAT) Terminology and Considerations*, 1999. [Online][Cited: 2017-03-16]https://tools.ietf.org/html/rfc2663.

[30] StatCounter. Desktop Operating System Market Share Worldwide, 2017. [Online][Cited: 2017-05-16] http://gs.statcounter.com/os-market-share/desktop/worldwide/#monthly-201704-201704-bar.

[31] StatCounter. Mobile Operating System Market Share Worldwide, 2017. [Online][Cited: 2017-05-16] http://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-201704-201704-bar.

[32] Statista. "Internet of Things (IoT) connected devices installed base worldwide from 2015 to 2025 (in billions)", 2017. [Online][Cited: 2017-05-14] https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/.

[33] Internet Live Stats. Internet Users, 2017. [Online][Cited: 2017-0514] http://www.internetlivestats.com/internet-users/#trend.

[34] Kenneth Straka and Gavin Manes. Passive detection of nat routers and client counting. In *IFIP International Conference on Digital Forensics*, pages 239–246. Springer, 2006.

[35] Wikipedia. Patch tuesday, 2017. [Online][Cited: 2017-05-12] https://en.wikipedia.org/wiki/Patch_Tuesday.

[36] Bo Zhang, Yangyang Guan, Wenjia Niu, Jianlong Tan, and Zhi Mao. A hybrid packet clustering approach for NAT host analysis. In *Communication Software and Networks (ICCSN), 2015 IEEE International Conference on*, pages 432–438. IEEE, 2015.