University of Twente
Enschede - The Netherlands

University of Twente
Department of Electrical Engineering, Mathematics an Computer Science
Chair for Design and Analysis of Communication Systems

Detecting UDP attacks in high speed networks using packet symmetry with only flow data

by

Daan van der Sanden

Master thesis
Executed from Januari 2008 to Juli 2008

Supervisors: Dr. ir. A. Pras
Dr. R. Sadre
A. Sperotto M.Sc.
Prof. dr. ir. B. Haverkort

Abstract

Attacks on the Internet are becoming a bigger problem since more users, companies and even complete societies rely on the correct functioning of the Internet. Some examples of these attacks are Denial of Service attacks or Port Scans. Most network operators deploy Network Intrusion Detection Systems to identify attacks and protect their selves against those attacks. Existing packet based inspection monitoring systems do not scale well with Gbps speed network technology. To overcome this problem, it is possible to use aggregated traffic information, like flow data as it is done in this thesis.

A tool, based on packet symmetry, was developed to detect attacks in UDP traffic. This tool was validated against a reference set. With the correct functioning tool, we were able to find three DoS attacks and numerous scans and other background noise in the data sets.

In this thesis, we show that, even for large Gbps networks, the symmetry in UDP packets using flow data is a good metric to detect attacks using UDP. Especially flooding attacks are easily detected. The analysis of two real high-speed networks with Gbps links, the University of Twente and SURFnet, supports this statement.

Acknowledgements

This thesis is the final part of my education and obtaining the title “ingenieur” (or Master of Science as it is called nowadays).There are a few people I would like to thank, that were of great help to me during this final assignment.

I would like to thank the ICTS and SURFnet for the supplied flow data that was analyzed. In particular I would like to thank Hans Trompert (SURFnet), Roel Hoek (ICTS) and Tiago Fioreze (PhD-student DACS) for helping me understanding the network layouts and settings of both the UT and SURFnet better so a meaningful analysis of the flow data sets was possible.

Secondly, the weekly DACS IDS meetings were of great help to me. Discussions ranging from database layouts to problems we all encountered analyzing the flow data were of great help for me. In this meeting, also my supervisors were present who I also would like to thank for their help with my thesis, in particular Anna Sperotto for helping me with my thesis to get it to the current level. I would like to thank Aiko Pras for his advice to get structure in my thesis and research. To make my stay in the DACS lab more enjoyable and for the sometimes-needed distractions during some of the long working days I would like to thank all students that were present in the DACS laboratory during my work on this thesis.

In a more general level, I express my gratitude for my parents, who supported me during my education that lasted a little longer than planned and always believe in me and allowed me to develop myself. Finally, I like to thank my girlfriend Susanne for getting my education in the final stage and keep me disciplined enough to go to the lab every morning, so I could finish this final assignment.

Daan van der Sanden

Contents

1 Introduction

The Internet is growing: more people subscribe via broadband connections and networks speeds are increasing. The backside of this growth is the increasing dependency on the Internet. Because of this dependency, the misuse of the Internet will have a larger impact on society. Recently a malfunction in the hardware of a Dutch network operator resulted in many companies disconnected from the internet for several days. Shopkeepers were for example unable to accept PIN-transaction but only cash [11].

Examples of this misuse or attacks are port scans and Denial of Service (DoS) attacks. One could also deliberately misuse the network for its own financial gain for example in the form of unsolicited e-mail (spam). To tackle these problems most network operators use Network Intrusion Detection Systems (NIDS) to detect those attacks on their network.

Because of the still growing demand of bandwidth, the availability of high-speed network equipment is also growing. Network speeds of 10Gbs are becoming normal these days. For the Network Intrusion Detection Systems currently in use, these speeds are a problem, because they are mainly based on deep packet inspection. This does not scale well to these high speeds. A more scalable way is to look at an aggregated version of this data, for example flow data, as in this thesis.

Most research today is focusing on detecting attacks in TCP connections using packet inspection. Not much research is done at anomalies in UDP traffic yet. Administrators in the field have the same feeling we had, that attackers nowadays use the UDP protocol more than the TCP protocol for DoS attacks. Therefore, the focus in this thesis is on detecting attacks in UDP traffic.

1.1 Research questions

This thesis answers the following research questions:

1. What kinds of attacks are described in literature and are detected on the Internet?
2. What detection methods can be used to detect (UDP) attacks?

The method that this thesis further investigates is based on packet symmetry of connections. This thesis investigates if it is suitable when only flow data are available (Chapter 3 will discuss the idea behind packet symmetry in more detail). In order to do this, the thesis will answer the following questions regarding this metric:

1. Is it (in principle) possible to use packet symmetry for detecting UDP attacks using only flow data?
2. At what granularities can packet symmetry be used to detect UDP attacks?
3. Is it also possible to use sampled flow data?
4. What UDP attacks can be found in real networks?

1.2 Research approach

To answer the first two questions this thesis presents the results of a literature study. The study describes some attacks that are described in literature and found on the Internet. In addition, some methods of detecting these attacks are presented in this thesis. One method of detecting attacks will be used to build a tool that works with only flow data.

To answer the other four questions regarding the packet symmetry, a tool was build that calculates the packet ratio from a flow data set. The tool will first be validated against a flow data set, in which we know there are several attacks present. After this validation, we will use the tool to analyze one week of flow data from two high-speed networks, one flow data set from the University of Twente (UT) and a sampled flow data set from the university Internet service provider SURFnet. The sampled set is used to answer the third question since almost all traffic from the UT goes through SURFnet. Only UT traffic is considerd to get a sampled flow set of all UT traffic. Which is compared to the original UT data set.

1.3 Structure of thesis

Chapter 2 presents the literature study. First, the kinds of attacks that are described in literature are presented. After that, some detection methods are presented. Chapter 3 will explain the idea behind the packet symmetry and why it can be used as a metric for detecting attacks. Chapter 4 describes the design of the tool that is based on packet symmetry in UDP traffic for legitimate traffic. Chapter 5 will discuss what analyses are performed with the designed tool. One of these analyses will then be used to validate the designed tool. Chapter 6 presents the result of the two dataset with flow data that are analyzed by the tool. Some attacks found in those dataset are discussed at the end of the Chapter. Finally, Chapter 7 presents our conclusions and future recommendations.

2 Background information

We live in an era in which most people, connected to the Internet, know what DoS-attacks are. They get pop-ups from their firewall that they are scanned and have to deal with spam on a daily basis. Many people had the "pleasure" to have a virus or some kind of malware installed on their computer. Although many of those subjects are related to each other, this sometimes results in the fact that the same terminology is used to describe different situation. To overcome these problems a short introduction of different attacks are given with the terms we will use in this thesis. This Chapter will give some information on the kind of attacks found on in literature and that were witnessed on the Internet. The second Section presents some methods to detect attacks.

2.1 Different attacks

This Section presents the most mentioned attacks in literature. It gives some structure to the different terms that can be found in literature. The following three attacks are described: attacks on the infrastructure, flooding attacks and scan attacks.

2.1.1 Infrastructure

Chakrabarti provided a taxonomy of attacks that compromises the architecture of the Internet in [5]. The paper describes four categories of attacks on the infrastructure of the Internet: DNS hacking, routing table poisoning, packet mistreatment and denial-of-service attacks.

The next Section discusses the latter in more detail. An end user cannot protect themselves against the first three attacks. At the moment, the correct functioning of these systems is based on trust between all network operators. When an attacker compromises one of their hosts or routers, he can misuse this trust to disrupt the correct functioning of the Internet.

An example of routing table poisoning was seen at the end of February 2008, when the Pakistani Telecom provider made it to the world news. They were trying to block YouTube in Pakistan after a court order, when they made a configuration error. Because of this error, YouTube was not only blocked in Pakistan but within minutes also in the rest of the world [18].

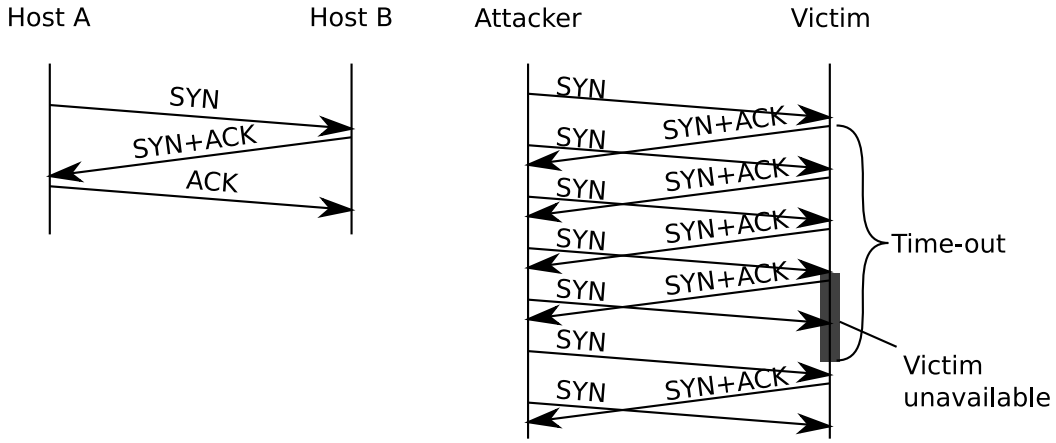


Figure 2.1: Legitimate TCP 3 way handshake and a TCP SYN attack.

2.1.2 Flooding

Flooding is often called a Denial of Service attack in literature. A Denial of Service (DoS) attack is defined by govcert.nl in [9] as an attack with the purpose to burden a system, service or network, until they are turned off or become unavailable. Therefore, a DoS-attack is much broader than just flooding a host and use up all his available bandwidth and/or resources. In this thesis, the focus will be on flooding attacks. Three often found and described flooding attacks are TCP-SYN attack, ICMP flood and UDP flood.

TCP-SYN

If we look at the TCP 3 way handshake, as shown on the left in Figure 2.1, that is used to establish a TCP connection, it can easily be used to attack a host. If an attacker start a connection by sending a SYN packet, but never sends the ACK packet after the received SYN ACK packet, the victim will wait for a certain time for the ACK packet. During this period, the host allocates resources while it is waiting for the reply. If the attacker sends a lot of SYN packets, the host may become unreachable, as shown on the right in Figure 2.1, if it exhausts the resources it has available for starting new connections.

ICMP flood

In this kind of attack, a large amount of ICMP packets is send to the victim. The effect of this attack can be multiplied by using miss-configured networks on the Internet. When an attacker spoofs the return address of the ICMP ping packet and replaces it with the address of the victim, and he sends it to the broadcast address of a network. When multiple hosts in that network respond to the ping request, all replies of the ping commands will go to victim, the network amplifies the attack since one packet will result in multiple packets towards the victim. Therefore, in total, the attacker can multiply its available bandwidth with the use of miss-configured networks; this is known as a SMURF attack. Most networks however cannot be used for this kind of attacks anymore, because most routers do not allow packets in their network destined to their broadcast address.

UDP flood

During a UDP flood, an attacker tries to use up all available bandwidth of the victim so that it becomes unavailable. In addition, a service on the host can crash by overwhelming it with information. The service cannot cope with all the available information and becomes unavailable for legitimate users. Miss-configured networks can also amplify a UDP flood, the same as with an ICMP flood; this is known as a FRAGGLE attack. There are also UDP attacks that were amplified by using DNS servers [22]. This method of amplification can be fixed by reconfiguring DNS servers.

DoS distribution

In [16] Moore et al. described a method to measure the number of Distributed Denial of Service (DDoS) attacks on the Internet that use IP spoofing. They use a network telescope to measure backscatter traffic. During three years of measurement, they were able to find some statistics such as duration and amount of DoS attacks that were initiated on the Internet. During these three years, they detected over 68 000 attacks. They also looked at what kind of attacks were performed, 95% of the attacks were done using TCP, 2,6% were ICMP attacks and only a small percentage of UDP floods was detected. The method that was used, is unable to detect reflective DDoS attacks and DDoS attacks done by botnets without any IP spoofing. Therefore, the number of attacks is higher than the numbers presented by [16].

2.1.3 Scans

Attackers use a port scan to see if a port on a network host is open. The goal is to see if the service that is listening is exploitable. Using errors in the services listening, an attacker could install software on this computer or steal information, whatever his goal is. Some scans that are described in the nmap [2] are TCP scan, SYN (or half open) scan, Stealth scans (like FIN scan, Xmas tree scan and NULL scan) and UDP scans. Since the main topic in this thesis is attacks in UDP traffic some extra information on the latter is given.

In UDP scans, it is harder to say for sure if a port is open or not, since UDP is a stateless protocol. So one should already know what service is listening, to make sure if a port is open by sending a good formatted packet to invoke a reaction. But because of the simple design of UDP an attacker uses the UDP protocol often to just send packets and they do not care if they arrive at a host or not. If they do, the attack was successful and the packet is accepted and processed. Examples of such behavior is spam [13] that (mis)uses the messenger service of the Windows operating system and the Slammer worm [15] that exploited a bug in the MSSQL service.

2.1.4 Distributed attacks

Section 2.1.2 described some commonly used floods. These floods were amplified most of the time because of miss-configured systems. Nowadays most networks are configured correctly so it is harder to misuse them for attacks. The threat now comes from another direction. Instead of miss-configured networks, one could also use high bandwidth attacks by using multiple hosts.

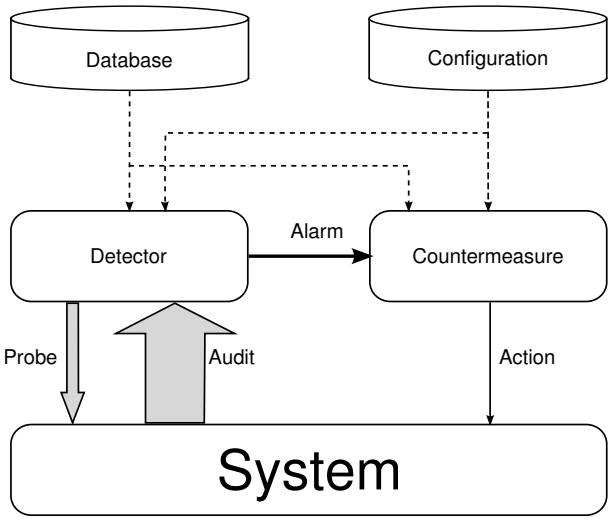


Figure 2.2: Basic detection system [7]. The width of the arrows represents the amount of data that is passed around.

When an attacker has the control of multiple hosts on the Internet, it can use all those hosts at the same time to initiate an attack. These hosts together are called a botnet. This access is usually gained using viruses, Trojans or social engineering to install software on the host. Known programs that give an attacker usage over multiple computer that do not belong to them are e.g. Trinoo and TFN(2K)[4].

Another possibility is in the form of a worm that infects hosts and starts an attack on a predefined time without any command or interference of the attacker. An example of this is the MyDoom worm that attacked SCO[23] and Microsoft[24] and as a side effect[8, 21] also made Google, AltaVista and Lycos unavailable for some time.

2.2 Detection methods

In order to protect systems against such attacks, operators use different detection methods. In Figure 2.2, a general representation is given on how such a detection system works. The system can be a single host or a complete network that needs protection. The audit data stream is the information that the detector analyses. The probe data is information that the detector can poll to help him with his decision to identify attackers. When the detector detects an attacks, based on the information from the database describing an attack and the configuration stating how sensitive the detector is, it raises an alarm. The IDS can take different action when it receives an alarm depending on the severity of the attacks and its configuration.

2.2.1 Firewall

To detect an attack on a host, several programs, implementing different methods, exist. The most known for end users are virus scanners and firewalls. If we look at the latter in more detail and look at it as represented in Figure 2.2 the system is a computer. The audit stream is all packets that are sent or received by the computer. The firewall inspects the

packets and matches the properties such as port number, source and destination addresses and application sending the packet to predifend rulesets that are stored in the database. Based on the settings in the configuration and filters stored in the database, alarms can be raised. If a packet does not follow any of the rules, the firewall drops the packet (action).

2.2.2 NIDS

A detection system that uses network traffic as audit data is called a Network Intrusion Detection System (NIDS). Therefore, if we map this to the general design displayed in Figure 2.2 the system is a complete network. The audit data can consist of all network traffic, packet headers or aggregate information from packet header. More detail on the audit data that can be used is in the next Section. The polling data for example can consist of SNMP data. In this section will discuss some research that is done on the design of the detector and complete NIDS systems.

Two NIDS that are freely available on the web are SNORT [19] and BRO [17]. Those NIDS packets scan packets and using different configuration files and plug-ins, so they can detect different kind of attacks. This is done by monitoring the content of packets while comparing it with signatures of known attacks. It is also possible to define suspicious behavior. If a host is connecting to every host on the network on the same port, it is probably a scanner. The alarms that are generated can be used to temporarily block hosts, based on the policy set.

Some research focuses on detecting attacks before they leave the network. The idea is that the NIDS detect compromised hosts by the way they behave. If they detect malicious traffic, it is delayed or blocked. In [14] Mirkovic defines three models of normal traffic. These three models return a number. If the number is above a certain value, the NIDS raises an alarm. For the UDP protocol, the paper looks at the following three thresholds: n_{conn} - upper bound on the number of allowed connections to a destination, p_{conn} - lower bound on the number of allowed packets per connection and UDP_{rate} - a maximum allowed sender rate per connection. In [12] Kreibich proposes a metric that uses the symmetry between sent and received packets in a connection. The tool that is described in this thesis is based on this principle.

2.3 Auditing the network

The audit information that the network sends can be different. This section describes two of those options. The first one is packet inspection. The second option is to look at aggregated packet information in the form of flow data.

2.3.1 Packet inspection

With deep packet inspection, a copy of every packet is sent to a system that analyses every packet it receives. Therefore, if a 10 Gbps links is analyzed the processing power of the system that is analyzing must have enough resources to handle all those packets. Specialized and expensive hardware is needed to cope with these speeds and traffic volume. If also the content of multiple packets is combined (for example inspect http traffic), it also needs to keep track of a big amount of sessions. Therefore, the scalability of these systems to cope with Gbps speeds is questionable.

2.3.2 Flow data

Instead of looking at all packets going through a network link, it is also possible to look at aggregated information of network traffic in the form of flow information. This way the NIDS only analyses a fraction of the original traffic. The *de facto* standard at the moment is NetFlow v5 and v9. The IETF IPFIX [10] working group is also working on a standard, which is based on NetFlow v9, to export flow information. The information about a flow is exported to a collector as records that are stored in UDP packet.

A flow is defined as a unidirectional stream of IP packets between a given source and a destination. Specifically, a flow is identified as the combination of the following five key fields: source IP address, destination IP address, source port number (if applicable), destination port number (if applicable) and the layer four protocol type. Together with this unique keys also other information can be stored for each flow, for example when the first or last packet were switched/routed, the number of packets and the number of bytes. It is also possible to store some routing information for each flow. To reduce the burden on a router, it is also possible to sample the packets. One out of every n packets will be considered for updating the flow information.

3 Packet symmetry

In the previous Chapter, several attacks and detection methods were discussed. Administrators these days have the feeling that attackers use the UDP protocol more frequently than the TCP protocol to carry out their DoS attacks. There is not much research done to UDP attack yet. This thesis will focus on detecting malicious traffic in UDP, such as UDP floods, scanning and misuse of vulnerable services.

In this Chapter, the idea behind packet symmetry and how it can be used to detect UDP attacks is explained. In the first Section, we discuss the general idea behind the packet symmetry and give some examples on the symmetry in UDP traffic when looking at applications that use UDP as their underlying protocol. In the second Section, the packet ratio is introduced. It is a way to give a mathematical representation of the packet symmetry.

3.1 The idea

UDP has a simple implementation as a result UDP is generally used for two types of applications: simple protocols/applications and application where small delays are necessary such as gaming and multimedia.

The first category is systems where UDP is used for protocols where a request in a single packet results in a response of one packet. An example is DNS where a host needs an IP-address of a certain hostname. Figure 3.1 shows a common DNS lookup for `www.example.com`. As can be seen for all hosts involved in this example, the number of sent and received packets are in balance. Another example is for file system support over a network (AFS, TFTP and NFS). In these cases, a UDP datagram is sent that requests certain blocks and the answer (the requested block) is given in one UDP datagram. This traffic is balanced. However, these systems are usually used for LANs only since they perform the best with small round trip times and without much packet loss.

In the second category, UDP is used for online gaming and multimedia applications, especially when a high delay is not desirable. For example, live video and/or audio streams, games and Voice over IP (VoIP). The first is not investigated in this thesis since multicast is usually used. For the second, the player playing the game needs feedback from the Internet how his action changed his environment in the game. It also needs to know what the players around him are doing and they need to know what he is doing, so there will be a certain

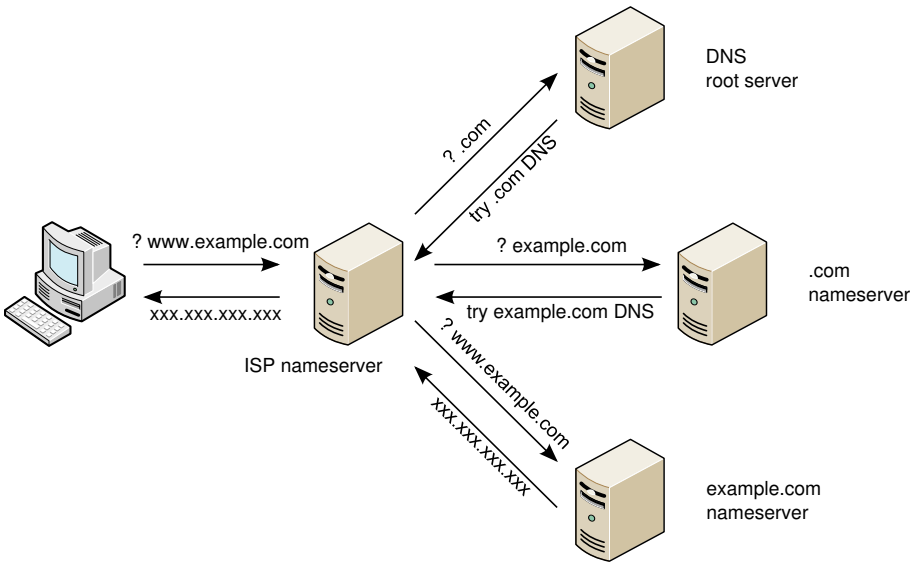


Figure 3.1: Example of a typical DNS lookup

balance.

For VoIP it is a little more complicated. UDP is usually also used for the initiation of the VoIP session, in those cases it depends on where the flow accounting takes place if the traffic is symmetric or not. When the session is setup, the symmetry of the stream of UDP packets is dependent on the exact compression and protocol used to transport the voice data. The behavior of the people participating in the call can also influence the number of packets sent and received. If a caller is talking more than he is listening, he always needs some feedback to know if the other side is still listening. With RTP, a RTCP stream is present that is balanced, so that could be used to distinguish a regular call from an attack.

3.2 Packet ratio

In [12] a method of selecting malicious traffic from legitimate traffic was proposed based on the packet symmetry. We will use the same metric to detect attacks in UDP traffic. Equation 3.1 is the metric proposed in [12], where tx is the number of transmitted packets and rx the number of received packets per unit of time. It is argued in this paper that within a legitimate flow the number of sent and received packets in a connection are balanced, S is equal to zero. When S is significantly different from zero, it is argued that the corresponding flow is malicious.

$$S = \log_e \left(\frac{tx + 1}{rx + 1} \right) \tag{3.1}$$

We will use a slightly different formulation but the basic principle stays the same: the number of transmitted and received packets should be in balance. Instead of using a logarithmic function where perfect symmetry in traffic will result in S having a value aroundThis is quite easy to do with our tool, since we also store the number of sent en received bytes. zero, we just look at the ratio between the incoming packets and the outgoing packets. Therefore, if

the number of incoming and received packets is in balance, the ratio will be one. Because we drop the logarithmic function, we do not need to add any numbers to the numerator or the denominator. This addition introduced a small error on the ratio of hosts not sending many packets. Since we are interested in attacks toward the network, we decided the received number of packets to be in the numerator instead of the denominator as presented in Equation 3.2. If the number of transmitted packets is zero, R_{UDP} (packet ratio) will be defined as a large number, M.

$$R_{UDP} = \begin{cases} \frac{rx}{tx} & tx > 0 \\ M & tx = 0 \end{cases} \tag{3.2}$$

This metric has one down side. When an attacker uses a host for a reflective attack and uses it as an amplifier the metric will not work. However, one should never rely on a single metric to identify attacks. An alternate metric that will work in the case of a reflective attack is the byte ratio as proposed in [20], which can be used together with the packet ratio.

In [12] multiple granularities were investigated and it was believed that host-to-host symmetry was a good starting point for further investigation. Since we are going to analyze the SURFnet traffic, we expect the number of host-to-host ratios to become too large to analyze them. We decided to analyze the traffic based on time intervals. The smallest time bin that we analyze, will be the same value as the active timer that was set during the capture of the flow data in the router. The active timer is the maximum duration of a flow when it is exported.

4 Tool design

In the previous Chapter, the idea behind packet symmetry was explained. In addition, a mathematical representation in the form of the UDP ratio was introduced. This idea was inplemented in a tool. The build tool is able to investigate all unicast UDP traffic. The tool we build stores all information about UDP traffic in a database for easier access and analysis.

The tool takes some simple steps to make it easier to evaluate the packet ratio (R_{UDP} in Equation 3.2) at host level:

1. Read and filter flow information.
2. Aggregate the available flows.
3. Store them in a database.
4. Visualize the data in the database.

The first step is quite straightforward, the exported flow information is processed and all unicast UDP traffic is filtered out and is processed by the next steps. The first Section describes the second step the tool takes. The second Section describes how the data is stored using a database. The last Section will explain how the ratios will be investigated.

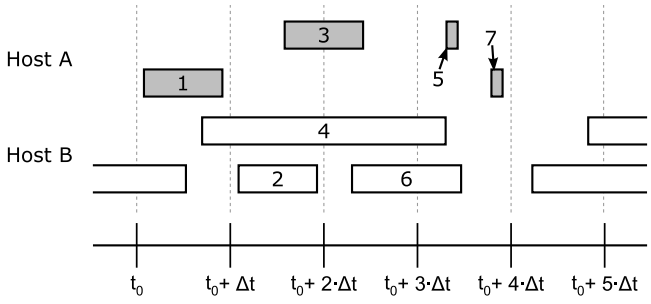
4.1 Aggregation

To reduce the number of states for the tool to tracks, the tool aggregates the flow information. Two types of aggregation are applied:

1. **Host aggregation**

   Since each flow is exported with the five unique keys that identify a flow, all traffic to and from a host arrives in multiple flow records[1]. So all flow records from and to a certain host are aggregated. The tool only uses the destination or source IP address to identify incoming or outgoing flow for the hosts for the monitored network. The port numbers of the flows are ignored.

[1] A flow can be split up in multiple flow-records, according to the exporting strategy of the router. In Appendix B more information on the exporting strategy of the router can be found.



(a)

flow number	packets	bytes
1	10	3000
2	15	4000
3	12	3500
4	30	5000
5	1	1500
6	45	4500
7	1	500

(b)

Figure 4.1: Aggregation of the NetFlow data, where (a) shows the timing behavior of a few flows and (b) shows the traffic information.

2. **Time aggregation**

   To reduce the number of states the tool needs to track, the tools stores the traffic information at fixed time bins in the database. The time bins are the same size as the maximum length[2] in a flow record.

Figure 4.1 illustrates a simple situation, which is used to explain how the tool aggregates the flows before they are stored in the database. The flow numbers indicate the order in which the flows are processed by the tool. All gray flows originate from host A and all white flows originate from host B. For simplicity, only one direction is explained, but everything also applies for incoming flows. In figure 4.1 t_0 is a UNIX timestamp in seconds and a multiple of the window size Δt.

When the tool encounters the first flow, it fits completely in one time bin and thus is stored as traffic for Host A at time t_0. The same is true for flow 2; it fits in one time bin and thus is stored as traffic for Host B for time t_0. When the tool processes flow 3, which length also would fit in one time bin, it is split up. The tool does this because the first packet was seen in a different time bin in which the last packet was seen. The tool will split the flow into two different parts. When the tool splits up the flow equally to the duration of the flow in each bin bins, it assumes that the packets and bytes were uniformly distributed during the duration of the flow.

Table 4.1 shows the result of all the flows processed by the tool for the example that is given in Figure 4.1. This process can also easily be applied for outgoing flows to host A or host B. The tool stores the following information in the database: UNIX timestamp of the time bin, host, incoming packets, incoming bytes, outgoing packets and incoming packets.

4.2 Database setup

The tool uses SQLite[3] as a database engine to store the data. SQLite is a serverless, zero-configuration SQL database engine. Because of these features, the data can easily be stored in one single file and we can use almost of all SQL statements to extract data.

[2]The length of a flow is the difference between the times the first and the last packet were switched or routed.

start time	host	packets	bytes
t_0	Host A	10	3000

start time	host	packets	bytes
t_0	Host A	10	3000
$t_0 + \Delta t$	Host B	10	3000

start time	host	packets	bytes
t_0	Host A	10	3000
$t_0 + \Delta t$	Host A	6	1750
$t_0 + \Delta t$	Host B	15	4000
$t_0 + 2 \cdot \Delta t$	Host A	6	1750

start time	host	packets	bytes
t_0	Host A	10	3000
t_0	Host B	3	500
$t_0 + \Delta t$	Host A	6	1750
$t_0 + \Delta t$	Host B	27	6000
$t_0 + 2 \cdot \Delta t$	Host A	6	1750
$t_0 + 2 \cdot \Delta t$	Host B	42	5000
$t_0 + 3 \cdot \Delta t$	Host A	2	2000
$t_0 + 3 \cdot \Delta t$	Host B	18	2000

Table 4.1: Stored aggregation tables after reading flow 1, 2, 3 and all flows from example illustrated in Figure 4.1

```
1  CREATE TABLE traffic (
       starttime       INTEGER,
       host            INTEGER,
       pckt_in         INTEGER,
5      byte_in         INTEGER,
       pckt_out        INTEGER,
       byte_out        INTEGER,
       UNIQUE ( starttime, host ) ON CONFLICT IGNORE
   );
10 CREATE INDEX traffic_idx ON traffic (starttime,host);
   CREATE TRIGGER trafficUnique BEFORE INSERT ON traffic
   WHEN EXISTS (SELECT starttime FROM traffic
                WHERE starttime=new.starttime AND host=new.host)
       BEGIN UPDATE traffic SET
15         pckt_in=pckt_in+new.pckt_in,
           byte_in=byte_in+new.byte_in,
           pckt_out=pckt_out+new.pckt_out,
           byte_out=byte_out+new.byte_out
       WHERE starttime=new.starttime AND host=new.host;
20 END;
```

Listing 4.1: SQL description of the used database

All the information we are storing is all ready explained in Section 4.1. The corresponding SQL description for creating the table is displayed in Listings 4.1 in line 1 to 9. The combination of the starttime and the host are unique. In line 10, an index is created for the starttime and host columns in order to speed up lookups for specific time bins and/or hosts.

Since the files are stored on the disk as PCAP files it is possible that data that from and to a certain host in the same time bin exists in two different files on the disk. Since the tool is quite simple in its design and setup, we let the database engine cope with this by adding a trigger. When a starttime and host combination all ready exist the new values are added to the ones already in the database instead of adding a new record with the same starttime host combination pair (lines 12 to 19 in Listings 4.1). The downside of this method is that when for some reason the process of filling the database is stopped it must be restarted completely. If this had not been done, some grouping would have been necessary when accessing the data, which would slow down it considerably.

Now that the necessary data are in the database to compute the different ratios, using the SQL query language it will be easy to extract it in the way we want it. For example,

if want to extract the packet ratio for the whole network per time bin, the following query can be used:

```
SELECT
      starttime,
      1.0*SUM(pckt_in)/SUM(pckt_out)
   FROM
      traffic
   GROUP BY
      starttime
   ORDER BY
      starttime ASC
```

4.3 Visualization

Instead of writing an algorithm to detect a possible attack, we visualize the data coming from the database in graphs and study the behavior of UDP traffic using a plot of the ratios. For this purpose, we use the gnuplot[1] program to make graphs out of the data extracted from the database. All the results in this thesis are plotted with the help of the gnuplot program. In all the graphs where we plot the packet ratio, a logarithmic scale will be used. This way when the number of received packets is twice the number of sent packets has the same distance in the graph to one as when the number of received packets is half the number of sent packets.

5 Analysis and Validation

The tool will be used to do different analyses on the data sets. Since we have an entry in our database for every time bin in which a host sent and/or received a UDP packet and an analysis time frame of one week, where the size of the time bin is in the order of magnitude of minutes, it will be infeasible to plot every host. Therefore, we have to group some data together to do our analyses. The analyses we will perform are described in the first section of this Chapter. In order to see if the designed tool from the previous Chapter is implemented correctly, we will discuss the result of the validation process in the second section of this Chapter.

5.1 Analysis

In this thesis, two different analyses are performed on the data sets: a time series analysis and a statistical analysis. In the first case, we will look how the packet ratio changes over time. This is always done on network aggregation level. Since we expect all hosts to have a packet ratio close to one, the whole network is also expected to have a packet ratio close to one. So all sent and received packets per time bin are summed up and divided to get the ratio for the whole network per time bin. This can be done with one query in our database. This type of analysis will also be used in the next Section to validate the correct implementation of the metric in our tool.

In order to investigate the effects of different granularities at which we calculate the packet ratio, these values are extracted from the database. The cumulative percentage of all those ratios will be presented and analyzed. We do this because it is impossible to plot a time series representation of all hosts in the database. To see what happens to all ratios that are calculated with the different granularities to investigate if the ratio is a useful metric, we plot the cumulative percentage. The two granularities that we are investigating will be the size of the time bins and the number of hosts that are aggregated together.

5.2 Validation

To test the validity of the implementation of the method in our tool, a dataset with flow data of the University of Twente, which was captured during another project, was analyzed.

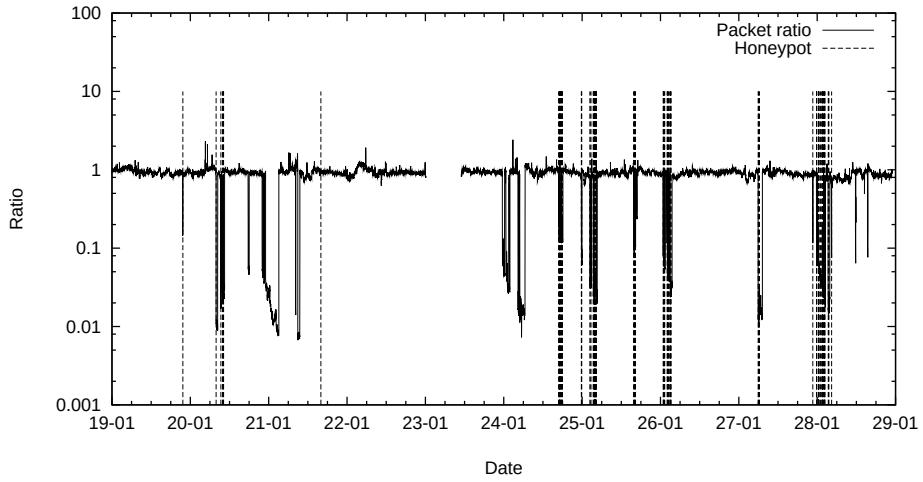


Figure 5.1: The ratio from the reference set over the whole period the honeypot was misused. The timestamps that were extracted from the logfiles of the honeypot are marked with a vertical dashed line. The gap at 23 Januari is because of hardware failure in the flow collector.

During this project a honeypot on the university was set up to monitor the behavior of "hackers". The honeypot was setup in such a way, that every log in attempt with a valid username and a password set (not empty) to the SSH server was successful. The activity was monitored by a simple patch of the SSH server, where every character displayed was logged with the timestamp when the text was displayed.

During this project the "hacked" honeypot was used for various UDP floods on several hosts outside of the university. In order to test our tool we match the timestamps of the log file from the honeypot with the results of a time series representation of the packet ratio, aggregated for the whole university. So every minute the ratio of all UDP packets is calculated.

The result of the time series analysis is displayed in Figure 5.1. We marked the times when an attack was started from the honeypot with a vertical, dashed line. Because of the way the honeypot was set up, the exact end times were not possible to determine. Figure 5.2 displays the packet ratio and the start times of the attacks from the honeypot during a small percentage of time.

The number of attacks that were initiated from the honeypot that were found in the log files was 68. With the designed tool using the packet ratio, 66 of the time bins in which the attacks started were marked as time bins in which an attack took place. The two attacks that were missed only lasted for several seconds. Since we were using time bins of one minute, those attacks did not contribute enough to disrupt the symmetry in UDP packets of the whole network during that time bin.

Figure 5.1 shows that there are still several attacks (that also originated from the honeypot) in the night of 21 Januari and 24 Januari that were detected by our tool, but that were not marked as start times of an attack according to the log file of the honeypot. At the moment of writing this thesis, the honeypot is still analyzed to see how these other attacks were initiated. Possible options are the use of installed bots on the honeypot.

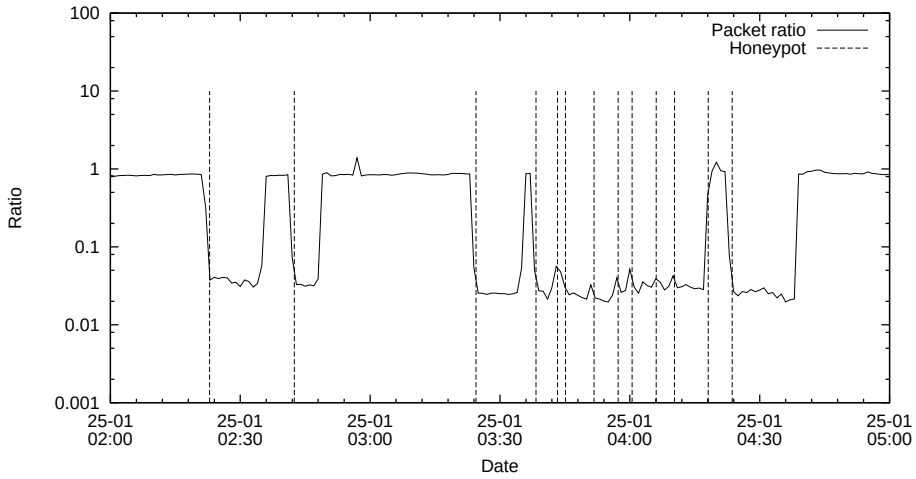


Figure 5.2: Zoomed version of the morning of Januari 1 of Figure 5.1 to show more precisely the match between the ratio and the start times of the attacks.

From this validation process, we can conclude that the tool implements the method, based on packet symmetry in UDP packets, correctly. The tool was able to detect 66 out of the 68 attacks initiated from the command line from the honeypot, and we detected some other attacks originating from the honeypot. So the tool can be used to investigate to detect UDP attacks.

6 Evaluation of the packet symmetry

Now that there is a validated tool, we can use this tool to analyze one week of real network traffic to investigate what kind of UDP attacks can be found. In this Chapter first describes the used measurement setup. Some statistics are presented to get an idea of the amount of information that needed to be processed. The second Section presents the time series analysis that was done on the networks from which we had flow data. The third Section presents the statistical analysis to get a better understanding of the granularities at which the packet symmetry is good measure. The fourth Section investigates and describes the found attacks in more detail.

6.1 Testbed

In this thesis, the tool analyzed flow data from two networks, the network of the University of Twente (UT) and SURFnet[1]. There was also the possibility to investigate the data from the GÉANT network, this was not used since the tool is based on the symmetry between sent and received packet, and the GÉANT[2] network is a transit network in where the reply of a connection might not go through this network at all. Because of this asymmetry our tool might quite easily give false positives. Figure 6.1 shows all these networks connect together. From all the gray routers the flow data is exported and stored. In Appendix A more information can be found regarding the internal layout of the network of SURFnet.

For the two networks that the tool will evaluate some settings of the router are important to know. As discussed in Section 4.1 the size of the time bins (Δt) needs to be determined. The size of the time bins will be the same as the maximum length the flows can have when they are exported by the routers. In other words, we choose the time bins to be the same size as the active timer of the router. For the routers of the University of Twente the active timer was set to 1 minute. For the SURFnet routers the active timer was set to 5 minutes. The SURFnet routers also had a sampling ratio of 1:100, which means that only 1 out of 100 packets are aggregated in the flow information.

[1] SURFnet is the academic research network of the Netherlands.

[2] The GÉANT project was a collaboration between 26 National Research and Education Networks representing 30 countries across Europe, the European Commission, and DANTE. Its principal purpose was to develop the GÉANT network - a multi-gigabit pan-European data communications network, reserved specifically for research and education use.

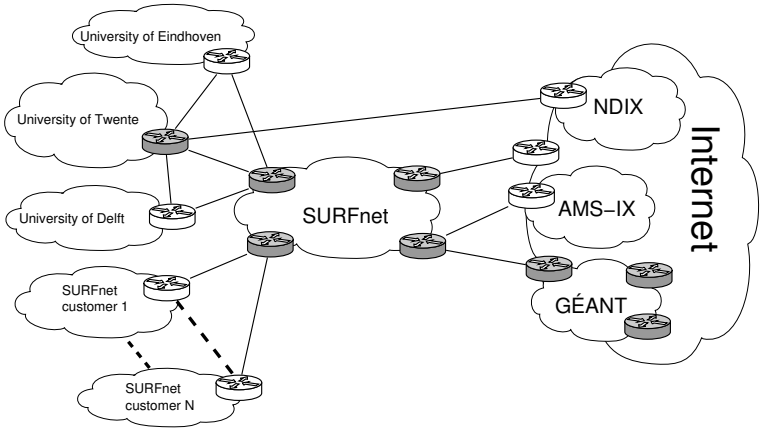
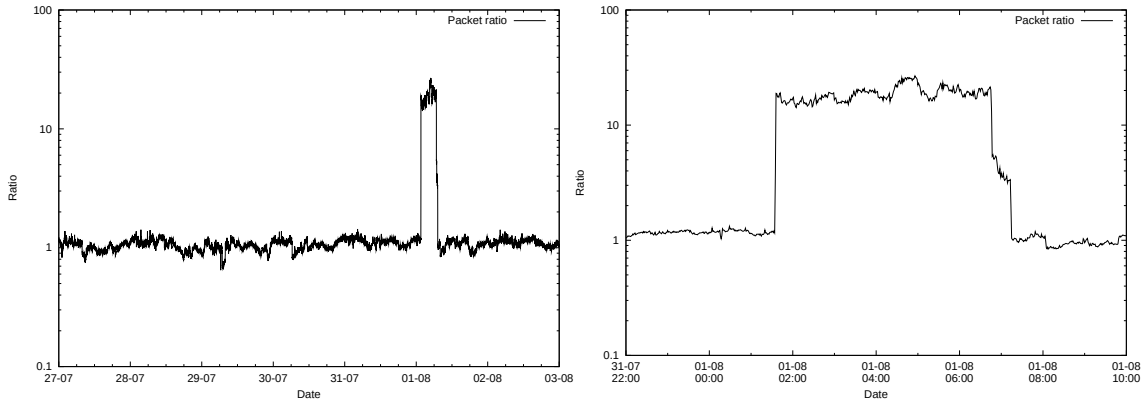


Figure 6.1: Setup that was used to gather the flow data. The gray routers are the routers configured to export the flow records.

	Sent packets	*Received packets*	*Sent bytes*	*Received bytes*	*Hosts*
UT - *total*	3 894M	4 850M	1 014 GB	587 GB	65 506
average	5 636 pps	7 020 pps	12 Mbps	7,0 Mbps	2 310*
SURFnet - *total*	38 080M	34 620M	14,1 TB	5,7 TB	3 367 464
average	55 212 pps	50 196 pps	171 Mbps	70 Mbps	9 531*

Table 6.1: The number of sent and received UDP packets and bytes during the one week of capture. (*) The average numbers are per time bin. In the case of the UT this 1 minute and for SURFnet 5 minutes.



(a) (b)

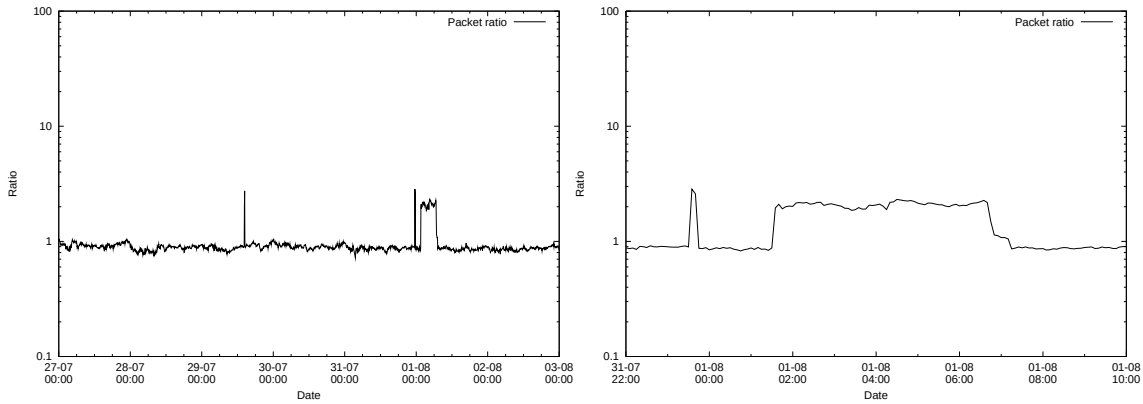
Figure 6.2: The time series representation of the whole UT network is plotted in (a). In the night of 1 August, an anomaly can be detected. This time region is enlarged in (b).

The flow data sets were captured over a period of one week, from 26 July 2007, 13:50 until 3 August 2007, 14:25. In Table 6.1 some information can be found about the two data sets. It lists per dataset all UDP traffic on the first line and the average values on the second line. The values of SURFnet are corrected for the sampling. All values were multiplied by hundred, except for the amount of hosts. In the rest of this chapter, an active host is a host that sent at least one packet in the time span we analyzed. In the case of the UT (a /16 address range) there were 12 553 active hosts out of the 65 506 hosts in the data set. Because of the sampling in the SURFnet data set, such numbers cannot be given with that precision for the SURFnet data set. If we would do the same analysis for all UT traffic in the SURFnet data set, 44 154 UT hosts were found. Of the 12 553 active hosts in the UT data set, only 10 517 were found in the SURFnet data set.

6.2 Time series analysis

The UDP packet ratio of the UT can be found in Figure 6.2. In Figure 6.2(b), a zoomed version of the ratio can be found of the night of 1 August 2007. It can be clearly seen that something suspicious was going on in that night. It appeared to be a UDP flood towards a UT host. More detail about the attack in the night of 1 August is given in Section 6.4.1. It can also be seen that the number of sent and received packets of the UT hosts are balanced around one. When we look at the time series representation of the SURFnet data, we expect the attacks also to be visible. As can be seen in Figure 6.3a this is the case. We can also see that there are two other SURFnet hosts under attacks from what appears to be flooding. More details about these attacks can be found in Section 6.4.1. We can also see that the ratio of SURFnet is slightly lower that one. Why this is the case is not exactly clear, but an explanation might be the effect of sampling.

If we only consider all entries in the SURFnet database belonging to UT hosts, we can investigate the effects of sampling on a flow data set. Because of the fact, that almost all traffic to and from the UT also goes through the SURFnet. This will result in a sampled



(a) (b)

Figure 6.3: The time series representation of the whole SURFnet network is plotted in (a). The attack on the UT is also clearly visable, also to other attacks are visible. One starting around 29 July 2007, 14:20 and the second attack starting around 31 July 2007, 23:35. In (b) is the night of 1 August enlarged.

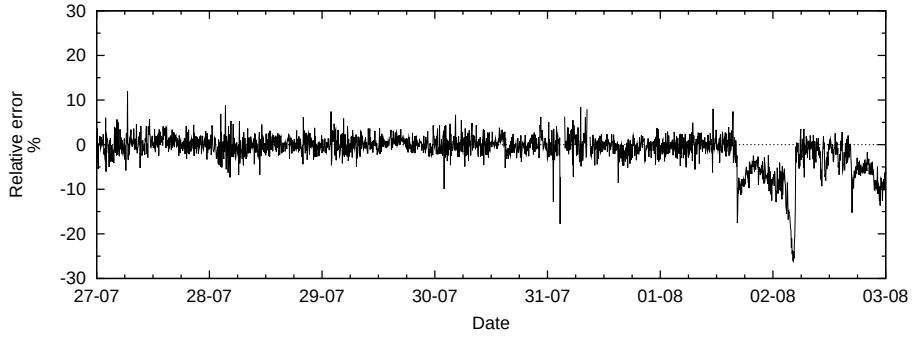
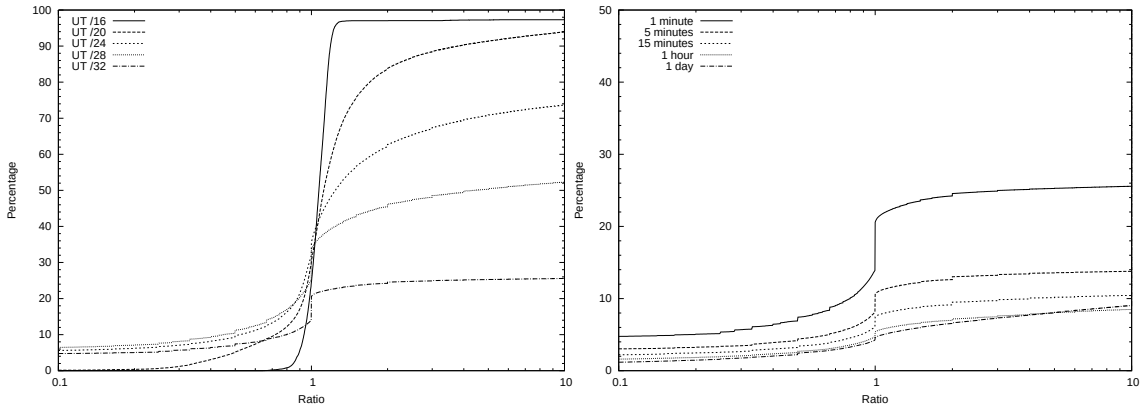


Figure 6.4: Comparing sampled ratio wih the unsampled version of the ratio.



(a) (b)

Figure 6.5: Cumulative percentage plots for the UT data set. In Figure (a) the granularity of the number of hosts we investigate was changed. In Figure (b) the granularity of the time bins was changed.

dataset of the UT. If we compare the ratios of the sampled UT set with the UT data set, we can see that they correspond quite well. In Figure 6.4, the relative error between the two ratios is plotted: most of the time the relative error is within a few percent. As can be seen at the end of the measurement period the relative error is larger, this is probably because of the fact that not all traffic is going through SURFnet.

6.3 Granularity research

This Section investigates how the tool, and thus the metric, performs at different granularities. In the previous Section and in Section 5.2 we could see that when we consider the whole network, the tool functions correctly at identifying flooding attacks. Now we will look at different granularities to see how it performs. Since it is unfeasible to plot time series plots for all hosts, we plot the Cumulative Percentage. In other words, the percentage of all ratios smaller than a certain value is plot.

Both the time and host granularity will be investigated in this section. The UT data set will be split in different subnets. The time bins will also be changed in size at host level granularity to see how the ratio performs under those conditions.

In Figure 6.5(a) it can be seen that when we use a host level granularity that more than 70% of the packet ratios of the UT data set are larger than 10. The UT has a /16 address range network. However, in our data set we only had 12,5 thousand active hosts. All inactive hosts were visited[3] (on average) 190 times that week. That means that 55% of the ratios larger than 10 at host granularity, were as a result of traffic destined for inactive hosts (because when an inactive host receives a packet, R_{UDP} will be defined as M). If we extrapolate this behavior of "background" noise to all hosts in the UT range, 47% of all ratios in the database with 1 minute time bins are as a result of this "background" noise. This can be assumed since only a few hundred hosts are active for the whole week; most of

[3]During one time bin of one minute, they received at least one packet.

Dataset	Filter (pckt/s)	Active host average	Packet-% (in/out)	Byte-% (in/out)
UT	all	2 310,19	100/100	100/100
	> 1	194,78	98,0/98,9	97,3/99,5
	> 10	64,88	91,9/92,0	89,5/95,4

Table 6.2: Effects of filtering: average amount of active hosts and number of packets are compared to the unfiltered data set.

the hosts are not active constantly. A more detailed insight on the mix of this "background" noise is given in Section 6.4.

This assumption also supported by the results in 6.5(b) where all distributed host ratios are plotted, but with different time bins. The larger the time bins the more ratios are larger than 10. With larger time bins the number of inactive hosts increases, but the number of ratios from legitimate hosts displaying normal balanced traffic will stay the same. This is not seen in the time series analysis because a few hosts are responsible for a great part of the number of packets.

To check this, a simple test was performed where we filtered out all low packet rate time bins, to get rid of most of the background noise. Some results of the resulting sets can be found in Table 6.3. As can be seen two filter criteria were tested. If the sending and receiving rate were below one pps, the host in that time bin was discarded. A second test was done where the limit was 10 pps. The filtered set behaves much better at host level granularity (see Figure 6.6).

6.4 Found attacks

In the last two sections, we saw that there is a lot of traffic to inactive hosts and UDP floods were performed towards the observed networks. To get a better understanding of what happened exactly both phenomena are investigated in more detail with the use of the original flow records. Both are described in more detail in this section.

6.4.1 DoS attacks

Three UDP floods were detected; two of them lasted less than 10 minutes. The one on the University of Twente lasted for a few hours. In all three floods, most traffic was generated with many small packets to port 53 (DNS). The attack towards the University of Twente consisted of 786M Packets and 39GB of data. 99,5% of the packets were exactly 46 bytes (88% of the total bandwidth consumption). The other traffic consisted of packets of 1500 bytes sent to all possible UDP ports. The first attack in the SURFnet data lasted for about 7 minutes and 76M packets of exactly 43 bytes (3.0 GB) were used. The second attack in the SURFnet data lasted for about 9 minutes and 49M packets of exactly 43 bytes (2.0 GB) were used. Both attacks show a great similarity in behavior, but the victims were different and the attacks also originated from different sources.

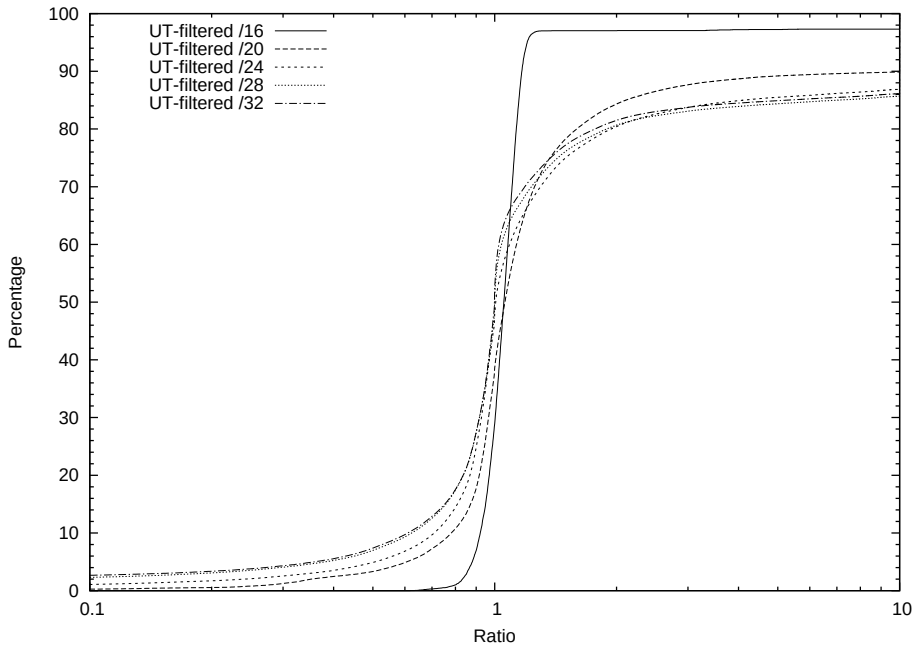


Figure 6.6: Cumulative percentage plot of all ratios in the filtered (> 1 pps) data set of the UT for different subnets.

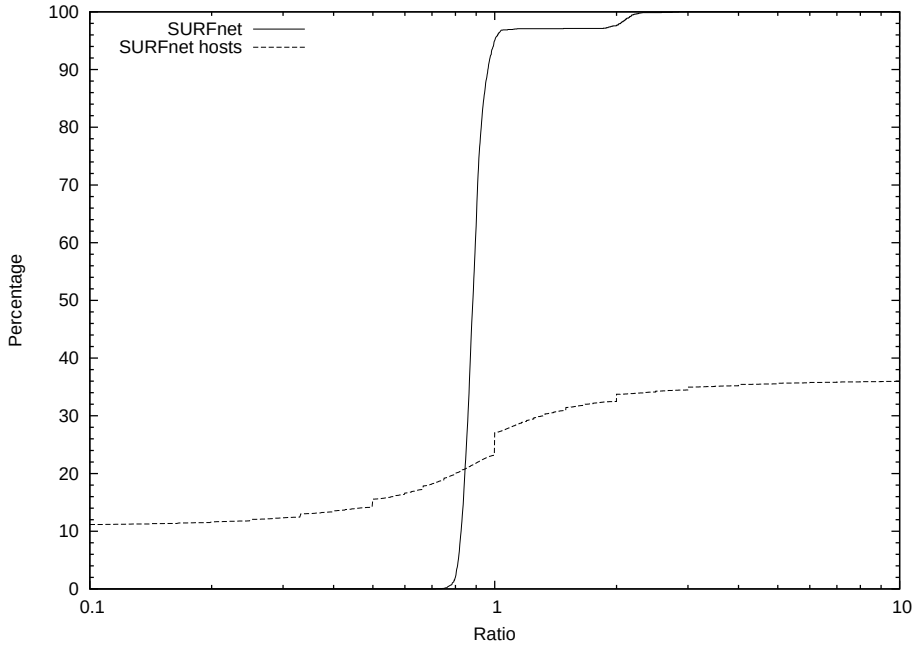


Figure 6.7: Cumulative percentage plot of all ratios in the data set of SURFnet at host granularity and the for the complete SURFnet network.

Port	*Packets*	*Percentage*	*Extra information*
1 026	2 555 326	22,6%	Messenger Spam
1 027	1 015 623	9,0%	Messenger Spam
6 167	670 547	5,9%	One single host as the destination address
16 443	447 835	4,0%	Two hosts as the destination address
11 795	248 561	2,2%	One single host as the destination address
27 015	203 722	1,8%	Half Life server
	6 182 911	54,5%	Other port numbers
Total	**11 324 525**	**100%**	

Table 6.3: Distribution of packets sent to inactive hosts in exactly two days.

6.4.2 UDP background traffic

In the previous Section, in all the cumulative percentage plots at host level granularity, a lot of traffic going to inactive hosts cause the plots not to distribute between 0 and 100 percent. A more thorough analysis was performed on the two days from the complete University of Twente dataset. A port distribution of UDP traffic sent to inactive hosts during a period of two days, From 1 August 2007, 0:00 until 2 August 2007, 23:59, was made. In this case, all inactive hosts were hosts that did not send any packet (no matter what protocol) during those two days. The result of this can be found in Table 6.3.

Popular ports are the ports 1026/1027 also known as the ports that messenger services uses, and is misused for displaying pop up messages on the screen of the user. The other top ports belong to one destination address per port (and in the case of port 16 443 two destination addresses). The connections came from many different hosts. On average around 100 different hosts per minute tried to communicate. When we look at the activity of that host over the whole week, it can be seen that in the beginning of the week the host was active (and the ratio was in balance), when the host was turned off, the packets kept coming. The numbers of packets gradually decreased over time. Because of the great number of different hosts sending packets we think this is from a peer-to-peer application running on the host.

7 Conclusions and recommendations

This chapter presents the conclusion of the resarch presented in this thesis. The conclusions of this thesis are the answers for the research questions stated in Section 1.1. All questions are repeated and answered. In the second section some recommendations and pointers for future research are given.

7.1 Research questions and conclusions

What kinds of attacks are described in literature and are detected on the Internet?

In Chapter 2 a short introduction was given on the different kind of attacks that can disrupt the Internet. The reader should have better idea on what can go wrong on the Internet. Although the description is not complete, it describes the most common attacks that are encountered so far. Section 2.1 describes some of those attacks, for the UDP protocol floods (or DoS attacks), worm outbreaks and scans are described.

What detection methods can be used to detect (UDP) attacks?

In order to protect the network against these attacks several detection systems were discussed in Section 2.2. We saw that those systems can rely on packet inspection (Section 2.2.2) or on flow data (Section 2.3). In addition, we saw some metrics that can be used to detect attacks. In this thesis in Chapter 3, we focused on the packet ratio to give a number to the symmetry in the number of packets in a connection. In this metric, we assume that the packets send via UDP are close or equal to the received packets. If this is not the case, something is going wrong.

Is it (in principle) possible to use packet symmetry for detecting UDP attacks using only flow data?

In order to detect the UDP attacks using only flow data, a tool was build (see Chapter 4) based on packet symmetry (see Section 3.2). The validity of the tool was tested using a reference data set with known attacks in them from a "hacked" honeypot. The developed tool was capable of detecting almost all floods out of a reference data set (see Chapter 5).

It was able to detect 66 out of the 68 attacks that were initiated from the honeypot. The two missed attacks were very short attacks. Because of our one-minute time bins, the tool was unable to detect them. Smaller t

At what granularities can packet symmetry be used to detect UDP attacks?

In Section 6.3 we looked at the cumulative percentage plots at host granularity of all ratios in the database. We could see that most of the time the UPD ratio at host level granularity was not balanced. After further analysis it was shown that is was mostly due to scans and or spam activity (small packets to inactive hosts with big intervals). This statement is supported by the result of the cumulative percentage plot of a filtered subset of the University of Twente (see Figure 6.6). In this filtered subset, we only analyzed hosts that sent or received more than one packet per second. Furthermore, this statement is also supported by the fact that when the time bins were enlarged the cumulative distribution plots got even worse. Because we investigated the behavior at host level granularity, the presence of this traffic was easily found.

For flood detection the best granularity level is to monitor bigger parts (or even the complete) network at once. With the current implementation of the tool and information we have, the method will not function on host level, because off the many inactive hosts that were contacted. When filtering was applied (and most of the ratios belonged to active hosts), the results were much better at host granularity.

For the time granularity, the choice of this value was chosen to be the value of the active timer in the router, the higher this value, the later a flow record is exported the later we know something is wrong. Smaller values are also possible, but when doing that no extra information is gained, besides a more sensitive ratio and more exact starting times can be estimated. Moreover, with smaller time bins in our tool we still have to wait for the timer in the router to expire before the data is available.

Is it also possible to use sampled flow data?

In both Section 6.2 and Section 6.3 we looked at the influence of sampling, by considering only traffic to and from the University of Twente out of the SURFnet data set and compared it with University of Twente data set. In the time series analysis of the entire University of Twente, we could see that influence of sampling was negligible.

However when we look at host granularity we can see some big differences. Some active hosts with small amounts of traffic are missed by SURFnet. As we saw in Section 6.1 16% of all active hosts in the UT data set is not found in the SURFnet data set. For all UT hosts in the UT data set 33% were missing in the SURFnet data set. This behavior of missing hosts and packets can be clearly seen in the different plots in Section 6.3. So sampled flow data will only give good results when used at larger granularities such as a complete network for flooding detection, otherwise sampling is not desirable for intrusion detection.

What UDP attacks can be found in real networks?

With the validated tool two data sets were evaluated, one from the University of Twente and SURFnet (see Section 6.1 for the layout). We were able to detect three flooding attacks (Section 6.2), one towards the University of Twente and two attacks towards other SURFnet customers. Section 6.4.1 showed that the main part of the attacks consisted of many small

UDP packets (43 or 46 bytes). The two attacks towards the SURFnet customers only lasted for a few minutes, the attacks on the University of Twente (which was clearly visible in both data sets) lasted for several hours.

In Section 6.4.2 a more detailed analysis was done, to see what the traffic existed in the background traffic. A lot of messenger spam was detected. We are not sure if it is still actively used or if it is a result of old viruses/worms that are still alive on the Internet. What appears to be some kind of peer-to-peer application also disturbs the ratio at host granularity a lot, because after the host appears to be turned off, many packets are still sent to the host.

7.2 Recommendations

For the granularity research, the data set was divided in different subnets. This was done on a mathematical basis. Another way would have been to divide them in to the logical parts of the researched network. In the case of the University of Twente, this could be based on the used vlans, or at a higher level split the network in to the student dormitories, servers and the different departments.

To do a better analysis at host granularity it is good to know when a host is active. Since when it is not active, all incoming traffic is more or less unwanted by definition. To better evaluate if the packet symmetry is a good measure to monitor individual hosts, it would be good to see how the ratio changes over time when they are active. This can be done using the DHCP server logs and the VPN access logs. It is also a possibility to develop a better algorithm to detect active hosts from the flow data. Instead of the complete dataset, smaller portions in time can be used. Also other protocols besides UDP can be used to see if a host is active.

Further research if this is a good measure for other protocols is also an option. The result presented by Kreibich [12] shows that there is also some symmetry flows from other protocols. If we look at TCP for example RFC 2581 (TCP Congestion Control) states:

> "The delayed ACK algorithm specified in [Bra89] SHOULD be used by a TCP receiver. When used, a TCP receiver MUST NOT excessively delay acknowledgments. Specifically, an ACK SHOULD be generated for at least every second full-sized segment, and MUST be generated within 500 ms of the arrival of the first unacknowledged packet."

Therefore, a symmetry of 1:2 between the number of received and sent packets is expected in a TCP stream. However, when the traffic is streaming in two directions instead of the situation described earlier where there is obvious client, the ACKs are piggybacked to the data and the ratio is expected to be close to one. Packet symmetry could also be a good measure in TCP streams and a ratio between 0,5 and 2,0 is expected.

It would also be interesting to investigate the packet symmetry using only flow data when look to it at a host-to-host granularity. This however will increase the number of entries that needs to be tracked. And especially when a host is under attack from a DDoS attack and with the use of large peer-to-peer networks, the number of states to track will become large. Or look at a smaller subset to see what happens.

Bibliography

[1] Gnuplot. http://www.gnuplot.info.

[2] nmap - network mapper. http://nmap.org.

[3] SQLite. http://www.sqlite.org.

[4] CERT. Distributed denial of service tools. http://www.cert.org/incident_notes/IN-99-07.html, November 1999.

[5] Anirban Chakrabarti and G. Manimaran. Internet infrastructure security: A taxonomy. *IEEE Network*, 16(6):13–21, November / December 2000.

[6] Cisco. Netflow solutions guide. http://www.cisco.com/univercd/cc/td/doc/cisintwk/intsolns/netflsol/nfwhite.pdf, January 2007.

[7] Hervé Debar. An introduction to intrusion-detection systems. In *Proceedings of Connect'2000*, May 2000.

[8] F-Secure. The side effect of Mydoom.M on Google, Yahoo, Lycos and Altavista. http://www.f-secure.com/weblog/archives/00000242.html.

[9] GOVCERT.NL. Aanbevelingen ter bescherming tegen denial-of-service-aanvallen. http://www.govcert.nl/download.html?f=59, February 2005.

[10] IP Flow Information Export Working Group. http://www.ietf.org/html.charters/ipx-charter.html, April 2008.

[11] NRC Handelsblad. Storing bij KPN treft internet en pinverkeer. http://www.nrc.nl/achtergrond/article1086758.ece, May 2008.

[12] C. Kreibich, A. Warfield, J. Crowcroft, S. Hand, and I. Pratt. Using packet symmetry to curtail malicious traffic. In *Proceedings of the Fourth Workshop on Hot Topics in Networks (HotNets-IV)*, November 2005.

[13] Microsoft. Messenger service window that contains an internet advertisement appears. http://support.microsoft.com/kb/330904, December 2007.

[14] J. Mirkovic, G. Prier, and P. Reiher. Attacking ddos at the source. In *Proc. 10th IEEE International Conference on Network Protocols*, pages 312–321, 2002.

[15] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver. Inside the slammer worm. *IEEE Security & Privacy*, 1(4):33–39, 2003.

[16] David Moore, Colleen Shannon, Douglas J. Brown, Geoffrey M. Voelker, and Stefan Savage. Inferring internet denial-of-service activity. *ACM Transactions on Computer Systems*, 24(2):115–139, 2006.

[17] Vern Paxson. Bro: a system for detecting network intruders in real-time. In *SSYM'98: Proceedings of the 7th conference on USENIX Security Symposium*, pages 3–3, 1998.

[18] RIPE RCC. YouTube Hijacking: A RIPE NCC RIS case study. `http://www.ripe.net/news/study-youtube-hijacking.html`, February 2008.

[19] Martin Roesch. Snort - lightweight intrusion detection for networks. In *LISA '99: Proceedings of the 13th USENIX conference on System administration*, pages 229–238, 1999.

[20] C. Siaterlis and B. Maglaris. Detecting ddos attacks with passive measurement based heuristics. In *Ninth International Symposium on Computers and Communications ISCC 2004*, volume 1, pages 339–344, 2004.

[21] Symantec. Symantec.com - Security Response - W2.Mydoom.M@mm. `http://www.symantec.com/security_response/writeup.jsp?docid=2004-072615-3527-99&tabid=2`.

[22] US-CERT. The continuing denial of service threat posed by DNS recursion (v2.0). `http://www.us-cert.gov/reading_room/DNS-recursion033006.pdf`, 2006.

[23] ZDNet.com. Mydoom downs sco site. `http://news.zdnet.com/2100-1009_22-5151572.html`.

[24] ZDNet.com. Mydoom variant targets microsoft. `http://news.zdnet.com/2100-1009-5149504.html`.

A

SURFnet layout

This Appendix describes some background information about the layout of the SURFnet network. This information is not part of the main research questions of this thesis, but in order to use the data from the SURFnet routers correctly some additional understanding about the topology is necessary.

As can be seen from Figure A, there are four routers that are exporting NetFlow records. Unfortunately, the records were send to one single port so that the records from all four different routers were in one file, and send to the same port, so the only difference was the UDP source port. However, since these could not be linked to the routers anymore we had to find a way to match the port number to a router.

The routers only exported flows going over outbound interfaces of the two different groups (edge routers and core routers). These interfaces are represented as black dots in Figure A. In the Figure can also be seen that the data from SURFnet needs to be split in at least two different datasets. Imagine a flow from one of the SURFnet customers to a non-SURFnet customer. This flow will pass two black dots going out of a region (the routers are grouped in pairs in two regions), which means it will be exported twice[1].

A small tool was created that resulted in a set of files, which listed information of the four routers. We use the UDP source port to identify the four different routers. The src_id field in the NetFlow header was stored for each of the four routers. Per src_id, all incoming and outgoing interface indices were saved. Per interface index, all Autonomous System numbers were stored. For the incoming interface, only the source AS was stored and for the outgoing interface index the destination AS was stored. These results were analyzed and based on how the as numbers were divided over the interfaces it was possible to detect which two routers were the edge routers and which two routers were the core routers.

A small snapshot of one of the exported files is attached at the end of this Chapter. In this snapshot, we see that the router sending from port 33920 and with src_id 83 has one input interface connected to networks with AS numbers belonging to SURFnet customers. The output interfaces connects to all kinds of different AS numbers all over the world. This is probably an edge router. In contrast with the core routers were the AS numbers of the SURFnet customers all have their own interface and the rest of the world is more grouped to one interface. To identify hosts that were located inside of the SURFnet we used a SNMP

[1]This is partially true. Since the routers use a 1:100 sampling there is a change that it will be exported. The more packets in a flow the bigger the change the flow will be exported twice

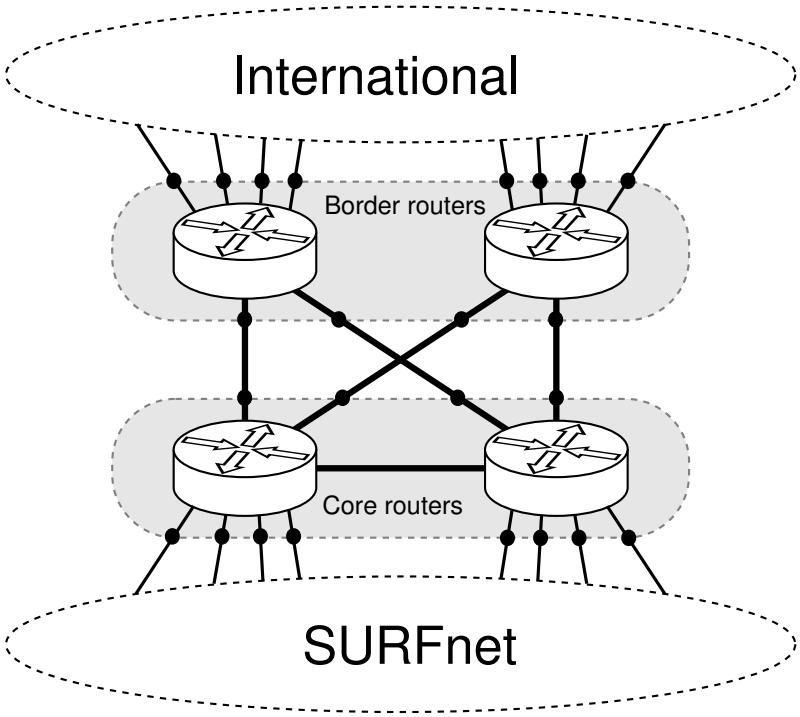


Figure A.1: Router layout SURFnet

walk of all the interfaces of all four routers. With the use of the interface index of a flow record, we were able to identify a flow as in incoming or an outgoing flow. This way we were able to store data from all hosts inside the SURFnet network.

```
From port 33920:
   Router with src_id 83:
      - Input snmps:
         -> 1644171827
            0,1101,1128,1132,1139,1161,20507,25182,29311,64560,64630,64690,64750,64800,64..
      - Output snmps:
         -> 0
            0
         -> 5308419
            46,70,87,131,137,159,224,237,239,278,291,297,302,372,376,378,513,553,557,559,..
         -> 5570562
            209,577,600,701,703,714,803,855,1239,1257,1313,1462,1540,1664,1668,1757,1761,..
         -> 5570566
            6746,6830,8404,8514,8751,9100,9141,13250,15547,15858,20877,21040,21466,24719,..
         -> 5636098
            558,1901,2571,3064,3228,3245,3257,3275,3313,3786,4058,4609,4662,4747,4750,47...
         -> 5636099
            1125
         -> 5636101
            9143
```

B

Cisco NetFlow

This appendix gives some more detailed information of NetFlow. However, this will also not be a complete description of NetFlow. With this Appendix, enough information is presented to understand better how implementation decisions were made during the research presented in this thesis. The first section will give a better understanding of the exporting strategy of flows of NetFlow enabled routers. The second section will go into more detail on what information was stored and exported with the NetFlow records exactly.

B.1 Exporting NetFlow records

The information of flows is stored in the NetFlow cache of the router. Since a router has a limited amount of memory, flow with their stored information can be exported. Another problem is that a flow needs to be exported before the 32-bit teller for the number of bytes is overflowing. When a flow is removed from the cache, it will be saved in a UDP datagram and send to the collector. There are four conditions when the router decides to export a flow from the NetFlow cache:

1. When the flow is idle for a certain time it will be exported. The name of the timer that keeps track of this value is called the inactive timer.

2. When the flow is active for a certain time it will be exported. The name of the timer that keeps track of this value is called the active timer.

3. When the cache is full. The router will export flows based on some algorithm.

4. In case of TCP, the flow will also expire when it encounters the end of a connection based on the set TCP-flags.

For the routers of the University of Twente, the exporting strategy is slightly different. The export of flows because of timers that expire are the same as described earlier, but the setting were these values are stored have slightly different names. At the University of Twente, the TCP flags are ignored and not exported. This means that the flow will not be exported when the TCP stream is ended with the appropriate TCP-flags set. So only the other three options remain.

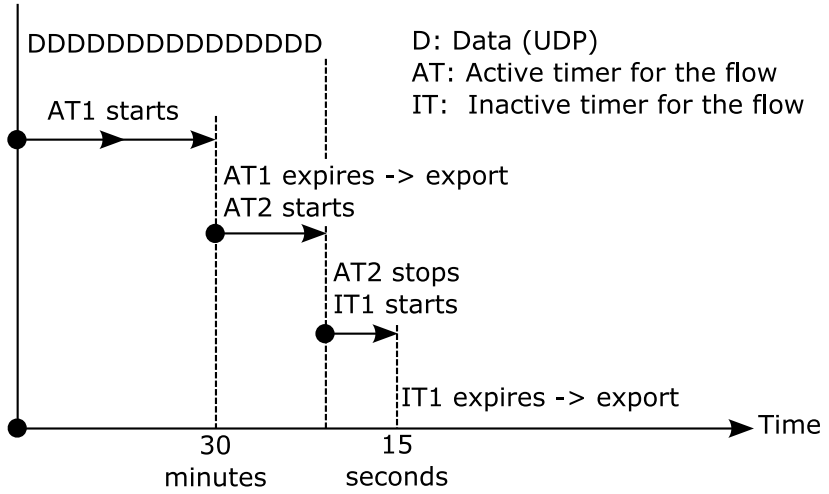


Figure B.1: Example of active and inactive timer.

To explain the use of the two timers (inactive/active timer), the example from the NetFlow Solutions Guide[6] is repeated here. In Figure B.1 a data stream longer than the value of the active timer (in this case 30 minutes) is sown. This flow will be exported twice, one time after 30 minutes and the second time 15 seconds after the end of the UDP data stream.

B.2 Export format

When flows are exported, they are collected together in a UDP datagram and sent to the collector. Before the records are included, also a header is included in the datagram. The exact layout of the UDP datagram for the various NetFlow versions can be found in the NetFlow solution guide [6]. In the next section, the information in the header will be explained, and the last section will describe which information is exported.

B.2.1 Header format

The NetFlow header of the exported UDP datagram consists on a number of fields, depending on the version. In Table B.2 and Table B.1 are the fields that are present in respectively NetFlow v5 and v9. The header is mainly used (by us) for two purposes, first to map the Sys uptime with the UNIX seconds timestamp, and secondly to see if all records were received by the collector. The timing fields of the headers are filled with their values at the moment the UDP datagram is exported.

B.2.2 Record format

The format of the records in the UDP datagram that is exported by the router with the flow records depends on the NetFlow version the device supports and is configured to use. In this thesis, both NetFlow v5 and NetFlow ve9 were used. NetFlow v5 uses a fixed layout for exporting the records which is define in the NetFlow Solution Guide [6]. In NetFlow

Field name	Description
Version	The version of NetFlow records exported in this packet (is always 9)
Count	Number of FlowSet records (both template and data) contained within this packet
System Uptime	Time in milliseconds since this device was first booted
UNIX seconds	Seconds since 0000 Coordinated Universal Time (UTC) 1970
Sequence number	Incremental sequence counter of all export packets sent by this export device. This value is cumulative, and it can be used to identify whether any export packets have been missed.
Source ID	The Source ID field is a 32-bit value that is used to guarantee uniqueness for all flows exported from a particular device. Collector devices should use the combination of the source IP address plus the Source ID field to associate an incoming NetFlow export packet with a unique instance of NetFlow on a particular device.

Table B.1: Header fields in NetFlow v9

Field name	Description
Version	The version of NetFlow records exported in this packet (is always 5)
Count	Number of FlowSet records (both template and data) contained within this packet
System Uptime	Time in milliseconds since this device was first booted
UNIX seconds	Seconds since 0000 Coordinated Universal Time (UTC) 1970
UNIX Nano seconds	Residual nanoseconds since 0000 UTC 1970
Sequence number	Sequence number of total flows seen
Engine Type	Type of flow switching engine, 0 for RP, 1 for VIP/LC
Engine ID	VIP or LC slot number of the flow switching engine

Table B.2: Header fields in NetFlow v5

Field name	Description
FIRST_SWITCHED	System uptime at which the first packet of this flow was switched.
LAST_SWITCHED	System uptime at which the last packet of this flow was switched.
IN_BYTES	Incoming counter for the number of bytes associated with an IP Flow.
IN_PKTS	Incoming counter for the packets of bytes associated with an IP Flow.
INPUT_SNMP	Input interface index.
OUTPUT_SNMP	Output interface index.
IPV4_SRC_ADDR	IPv4 source address.
IPV4_DST_ADDR	IPv4 destination address.
PROTOCOL	IP protocol byte.
SRC_TOS	Type of Service byte setting when entering incoming interface.
L4_SRC_PORT	TCP/UDP source port number i.e.: FTP, Telnet, or equivalent.
L4_DST_PORT	TCP/UDP destination port number i.e.: FTP, Telnet, or equivalent.
IPV4_NEXT_HOP	IPv4 address of next-hop router.
DST_AS	Destination BGP autonomous system number.
SRC_AS	Source BGP autonomous system number.
DST_MASK	The number of contiguous bits in the destination address subnet mask i.e.: the submask in slash notation.
SRC_MASK	The number of contiguous bits in the source address subnet mask i.e.: the submask in slash notation.
TCP_FLAGS	Cumulative of all the TCP flags seen for this flow.

Table B.3: Used fields in the SURFnet NetFlow v9 records

v9 however, the user can configure the router to export records in a dynamic way. It has to define a template that describes all the information that needs to be exported. In our case, the template that was used by NetFlow v9 to export the flow information contained the same information as NetFlow v5. The NetFlow v9 fields (and thus also the NetFlow v5 fields) that are exported can be found in Table B.3. A small description of each field is given in the table.