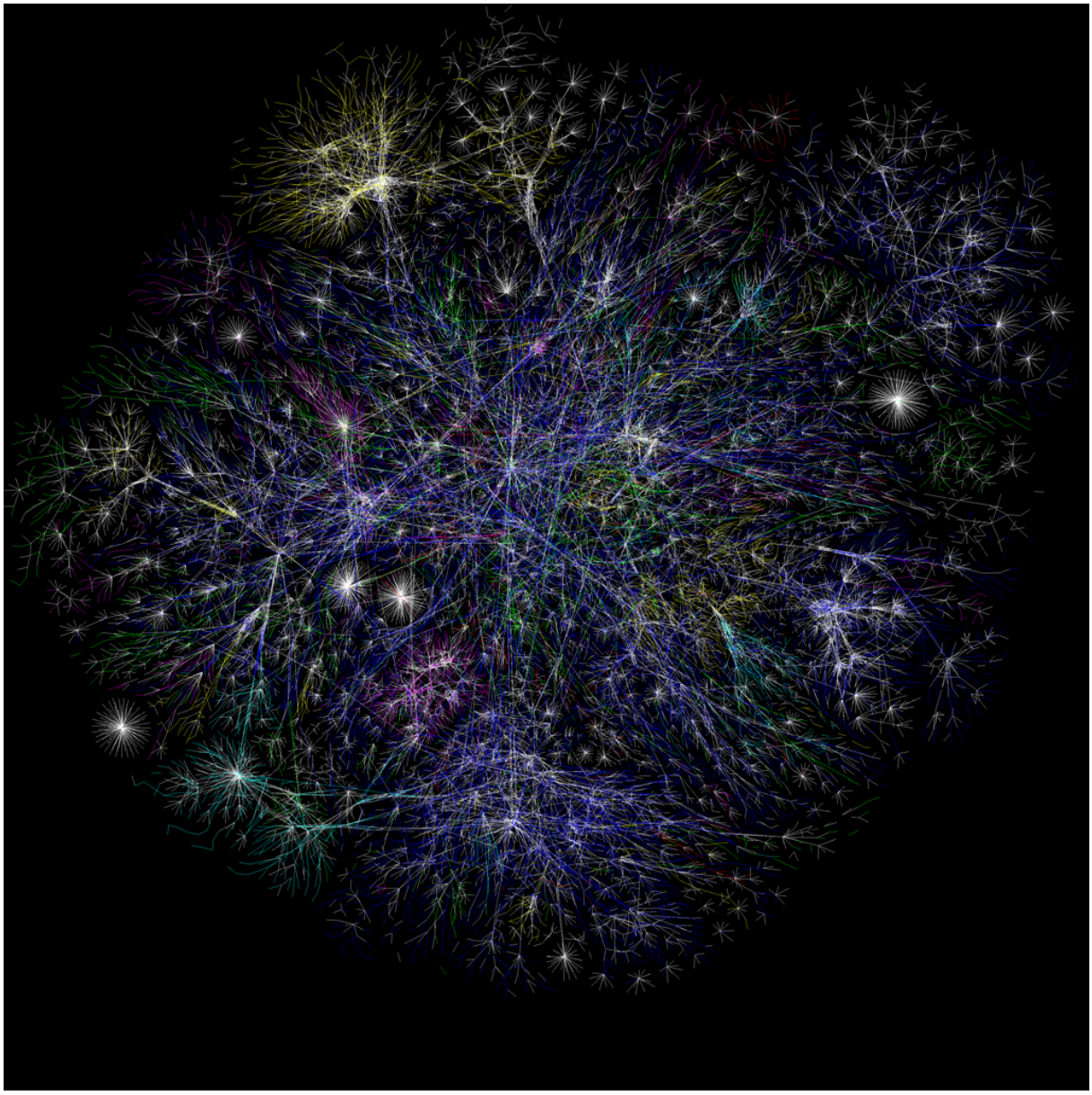
Detecting attacks involving DNS servers

A Netflow data based approach

Stephan Roolvink



Supervisors: Dr. ir. A. Pras
Dr. R. Sadre
A. Sperotto M.Sc.

University of Twente
Enschede - The Netherlands

Detecting attacks involving DNS servers

A Netflow data based approach

Stephan Roolvink

Supervisors:
Dr. ir. A. Pras
Dr. R. Sadre
A. Sperotto M.Sc.

Source frontpage image: vnunet

University of Twente
Enschede - The Netherlands

Abstract

The number of attacks on Internet services has been on the rise since the Internet became available to the general public. One of the services that has also been attacked using various ways is the Domain Name System (DNS) service. The DNS is one of the most important parts of the Internet. Without it, people would not be able to connect to favorite websites or check their e-mail. New attacks on services appear almost daily, like the DNS cache poisoning attack that was discovered by Dan Kaminsky. To defend against such attacks, a number of approaches have been researched and implemented, with varying success.

In this thesis several steps were taken to gain insight into the DNS service and the attacks that plague them. Log files from two DNS servers was used to gain insight into the DNS traffic that client and the DNS servers generate. The different types of attacks that are possible were analysis and described. The detection of attacks is done by describing the characteristics of the attacks and deriving methods of detection. One of these methods, called the relative entropy method, was tested in an effort to validate the effectiveness of this method. The goal that will be presented in the thesis focuses on detecting attacks involving DNS servers, using high level flow data gathered at routers.

From the research it could be concluded that the implementation of many DNS clients and the DNS server software BIND have implementation issues that need to be dealt with. The research has also shown that several different types of attacks exist that threaten the DNS service. From the research it could be concluded that certain attacks can be found using only Netflow data. The use of relative entropy method has shown that with more research can be used to detect certain attacks.

Preface

The thesis that you are about to read is the result of final masters project done on the University of Twente at the Design and Analysis of Communication Systems (DACS) chair. In a lab on the fifth floor I spend my time "quietly" trying to solve the problems of detect DNS attacks with only Netflow data. The research enticed me to use statistical measures I had not even heard of before the start of the project, which was hard but interesting to do. The literature about DNS traffic showed me many aspects on DNS traffic that were intriguing to know. It was great fun working on this project as it entailed a lot of thinking and doing stuff.

I want to thank Aiko Pras, Anna Sperotto and Ramin Sadre for their guidance during the research. The guys at ICTS for their help in gathering the data that was used in the research and their insight into the structure of the network.

This thesis would never ever have seen completion if my father and mother would not have supported and trusted me like they did (and the occasional needed kick in the butt). This thesis would be in much poorer state, if my girlfriend Désirée Pikaar and my fellow students and friends Jochem Rutgers and Martijn van Eenennaam, had not helped me out with improving the grammar, spelling and content in this thesis. And I would like to thank Gert Vliek for being the bringer of the end of the week (he always did that and nobody knows why).

Finally I want to thank Jochem Rutgers, Martijn van Eenennaam and Gerwin Ruiterkamp for their bottles of wine during the Christmas dinner on 20 December 2008.

I liked writing this thesis, so I hope you like reading it, it might be a little rough but hold interesting information.

Stephan Roolvink
Enschede, December 15, 2008

Contents

Abstract **v**

Preface **vii**

1 Introduction **1**
1.1 The research . . . 1
1.2 Research approach . . . 2
1.3 Outline . . . 3

2 State of the art **5**
2.1 The DNS protocol . . . 5
2.1.1 DNS response codes . . . 8
2.1.2 Extensions to DNS . . . 9
2.2 Network Intrusion Detection Systems . . . 12
2.2.1 Packet symmetry . . . 12
2.2.2 Cache poisoning detector . . . 12
2.2.3 Tunneling attack detector . . . 12
2.3 The Netflow data . . . 13
2.4 The TCPdump data . . . 14

3 DNS statistics **17**
3.1 Netflow data statistics . . . 17
3.2 TCPdump traffic statistics . . . 19
3.2.1 Measures . . . 19
3.2.2 Outcome . . . 21
3.3 Summary . . . 28

4 Attack overview **29**
4.1 Types of DoS attacks . . . 29
4.2 Attacks targeting DNS servers . . . 31
4.3 Attacks using DNS servers . . . 34
4.4 Traffic characteristics of the attacks . . . 34
4.4.1 DoS attack . . . 35
4.4.2 Botnet DDoS attack . . . 35

4.4.3 Reflection DDoS attack . . . 36
4.4.4 Recursive query attack . . . 36
4.4.5 DNS cache poisoning . . . 37
4.4.6 Buffer overflow . . . 37
4.4.7 Port scans . . . 37
4.4.8 DNS tunneling . . . 38

5 Search methods for finding attacks **39**
5.1 Search method per attack characteristic . . . 39
5.1.1 DoS attack . . . 39
5.1.2 Botnet DDoS attack . . . 39
5.1.3 Reflection DDoS attack . . . 40
5.1.4 Recursive query attack . . . 40
5.1.5 DNS cache poisoning . . . 40
5.1.6 Buffer overflow . . . 41
5.1.7 Port scans . . . 41
5.1.8 DNS tunneling . . . 41
5.2 General detection methods . . . 41
5.2.1 Packet per flow method . . . 41
5.2.2 Average number of connecting hosts . . . 45
5.2.3 Average number of flows . . . 46

6 Validation **49**
6.1 Relative entropy validation . . . 49
6.1.1 Relative entropy calculator . . . 49
6.1.2 Relative entropy output . . . 52
6.1.3 Threshold calculation . . . 60
6.2 Summary . . . 62

7 Conclusions and Recommendations **63**
7.1 Conclusions . . . 63
7.2 Future work . . . 65

Appendices **69**

A DNS requests and their responses **69**
A.1 UDP traffic . . . 69
A.2 TCP traffic . . . 70

B Standard deviation **75**

List of Figures

2.1 DNS message sequence: recursive querying 6
2.2 DNS message sequence: iterative querying 6
2.3 DNS response for www.utwente.nl (using DIG) 7
2.4 Small example network 8
2.5 Zones in DNS hierarchy 10
2.6 AXFR zone information example 10
2.7 Network topology overview 14
2.8 TCPdump capture mirror port setup 15

3.1 Source port distribution DNS clients 26
3.2 Source port usage Restricted range DNS clients 26
3.3 Source port distribution UT DNS servers 26
3.4 Query ID distribution DNS client 27
3.5 Query ID distribution UT DNS servers 27

4.1 The IPv4 header 30
4.2 A DDoS attack using Botnets 30
4.3 A DoS attack. 31
4.4 A Reflection DDoS attack 32
4.5 Cache poisoning — Dan Kaminsky 33
4.6 Port scan features 38

5.1 Distribution of average packets per flow — UT dataset. 42
5.2 Standard deviation packets per flow — UT dataset. 43
5.3 Skewness illustration (source Wikipedia) 44
5.4 Two identical probability distributions — with Relative entropy zero 45
5.5 Two different probability distributions 46
5.6 Distribution of connecting hosts 47

6.1 Time plots of two hosts with flows having the highest average packets per flow . 50
6.2 Relative entropy output UT DNS servers ($gc = 0.001$) 53
6.3 Relative entropy output UT DNS servers ($gc = 1^{-100}$) 54
6.4 Distribution comparisons 55
6.5 Distribution comparison — Er=0.228 56

6.6 Probability impact 56
6.7 Relative entropy modifications 57
6.8 Relative entropy — probability domain separation 57
6.9 Relative entropy — Bin merging 58
6.10 Distribution comparison — Re=0.015 59
6.11 Relative entropy — i impact 59
6.12 Relative entropy — Remove top packet per flow bin 60

A.1 Sequence diagram of DNS connection over TCP 71
A.2 The IPv4 header 71
A.3 Packet size DNS distributions 73

List of Tables

2.1 Options Resource Record 11

3.1 Balance of DNS servers against active hosts 18
3.2 Balance of TCP versus UDP DNS traffic 18
3.3 The number of UDP DNS servers of the total amount of DNS servers 18
3.4 Amount of DNS traffic compared to total traffic 19
3.5 Client to DNS server statistics 22
3.6 DNS server recursion statistics 23
3.7 Top 10: Requested top level domains 24
3.8 Top 10: Requested unknown TLDs 24

6.1 Distribution of packets per flow 52
6.2 Example distributions — High entropy but no attack 54
6.3 Example distribution — Domain separation 58
6.4 Lower and upper bounds for Relative entropy threshold 60

A.1 The TCP flags of request and corresponding response using BIND 72

B.1 Statistics of the top 30 hosts 76

1 Introduction

Over the years the internet has expanded to enormous proportions, with increasing numbers of hosts and availability of high-speed connections. This expansion has also lead to an increase in the number of attacks on hosts. The Domain Name System (DNS) is one of the important parts of the internet. Without it most people would not be able to connect to their favorite website. It is not hard to imagine that DNS servers have also been the targets of attacks. These attacks are possible because of exploits in the DNS protocol or bugs in the DNS software. There is also a group of attacks that overload a host with packets taking up massive amount of bandwidth and processing power in the hope of making the DNS server unavailable for genuine users. These attacks are called Denial of Service (DoS) attacks. The reasons for these attacks differ: for example a DNS cache poisoning attack is used to get control over a domain, while DoS attacks just want to disrupt normal service. Although system administrators are continuously adding new lines of defense to protect their infrastructure, the new generation of attackers is continuously trying new approaches to find ways to circumvent these defenses. Recently there have been reports of domain hijacks of several Internet Corporation for Assigned Names and Numbers (ICANN[1]) addresses [57]. This raises a question: if a well maintained organization like ICANN can be victim of a domain hijack, which other organizations have been victims of similar attacks? The use of high-speed networks makes low-level packet inspection a costly operation, so new ways are needed to perform traffic analysis for Intrusion Detection Systems (IDS). For this project, Netflow is used to perform traffic analysis. Netflow data is aggregated flow level data that was captured at routers.

1.1 The research

The use of Netflow data for high-speed traffic analysis would put less strain on a Intrusion Detection System. However the use of Netflow comes at the cost that the packets have been stripped of their content. As Sir Francis Bacon once said: “Knowledge is power” [4]. With this loss of the packet content, some attacks will be impossible to find, because in those cases only the packet content contains relevant information. This leads to the following research question:

How can attacks on DNS servers be detected by using Netflow data from routers?

In order to completely answer the main research question, a few sub questions are formulated. Each of these sub questions is part of the main research question and focuses on a certain aspect of the main research question. The sub research question that where formulated are:

[1] ICANN is a nonprofit organization that assign domain names and IP addresses [25].

1. **What are the types of attacks that target DNS servers, and in what way?**

   Before it is possible to detect attacks is best to know the possible attacks and their characteristics.

2. **What are the traffic characteristics of the attacks and how do they translate to Netflow data?**

   The possible attacks have characteristics that might be found in Netflow data, but it might also have characteristics that cannot be detected. For example: a traffic characteristic can be that during an attack the ratio of large packets from the DNS is larger than normally expected. The use of Netflow data might make it impossible to use certain traffic characteristics to find attack. This means that information needs to be identified that can be used to construct a viable way to look for attacks.

3. **Which algorithms can be used to successfully detect attacks?**

   Using specific detection methods for every attack leads to more overhead than necessary. If attacks have overlapping characteristics, one detection method could be used to detect both attacks.

4. **What are the false positives probabilities of the algorithms that can be used?**

   Every search algorithm will have a probability that it will give false positives on the existence of an attack [62]. This means that the method says there is an attack, which was not the case.

5. **What are the false negatives probabilities of the algorithms that can be used?**

   Every search algorithm will also have a probability that it will give false negatives on the existence of an attack [62]. This means that the method says there is not an attack, which there actually was.

1.2 Research approach

The following approach is used in this thesis.

- **Literature study**: From literature, information is gathered about the workings of the DNS protocol, the possible attack and their characteristics. The literature study is used to answer sub research questions one and two. Sub research question one is answered in chapter 4 and sub research question two is answered in chapter 5.
- **Gathering statistics**: The data used in the project contains information about many different aspects of the usage the DNS protocol. A statistical analysis is performed to get a better insight into the usage of the DNS protocol and helps to understand certain flows, that can be found in in the Netflow data.
- **Analysis**: The characteristics that were found in the literature study need to be linked to a detection method for Netflow, if that is possible. The methods that were described for the attack characteristics are analyzed to find more general methods of finding attacks. This part of the approach should help answer the sub research question three. This sub research question is answered in chapter 5.
- **Validation**: Validation of the usefulness and accuracy of the methods of the previous step is done by using the Netflow data. The validation is done to answer the sub research questions four and five. These questions are answered in chapter 6.

1.3 Outline

Chapter 2 describes the state of the art of DNS servers, while chapter 3 describes various statistics about the DNS protocol usage that was extracted from the data that was used for this thesis. Chapter 4 discusses various attacks and corresponding traffic characteristics. Chapter 5 explains the possible search methods for the attack characteristics and also gives more generic methods of finding attacks. Chapter 6 shows the validation of one the search methods that was described in chapter 5. Chapter 7 is used to draw conclusions on the research questions.

2 State of the art

In this chapter the state of art will be described of the DNS protocol and Network Intrusion Detection Systems (NIDS). Also a description will be given of the Netflow information and it database as it was used in this assignment.

2.1 The DNS protocol

A DNS server is used to translate human readable domain name to the corresponding IP address. The basic scenario of a DNS resolution can be seen in figure 2.1. A sequence of steps is taken to complete the address resolution: in the situation depicted in figure 2.1 the DNS server has an empty cache (see below).

1. A client asks its local DNS server for an address resolution.
2. The local DNS server asks the root DNS server for the address resolution.
3. The root DNS server responds with a referral to the top level DNS server.
4. The local DNS server asks the top level DNS server for the address resolution.
5. The top level DNS server will respond with a referral to the second level DNS server.
6. The local DNS server asks the second level DNS server for the address resolution.
7. If this second level DNS server is the authoritative DNS server for the queried address, then it will respond with the IP address of the host or an error. Otherwise it will respond with the address to the third level DNS server.
8. The local DNS server responds to the client with the answer to its query.

In total there can be 127 levels of DNS servers, so a search can continue for several more requests. The local DNS server will store the results of this query in its cache so that if the address is asked again it can give a faster response. The DNS server described above is a recursive DNS server, for it recursively queries DNS servers until it finds the authoritative DNS in which the search host should reside and then it responds. An alternative to recursive querying is iterative querying. With iterative querying the client will do the entire DNS address resolution, instead of the local DNS server. Figure 2.2 shows the sequence of events of an iterative querying client. It can be seen that the first step taken in this iterative search is a query to the local DNS server.

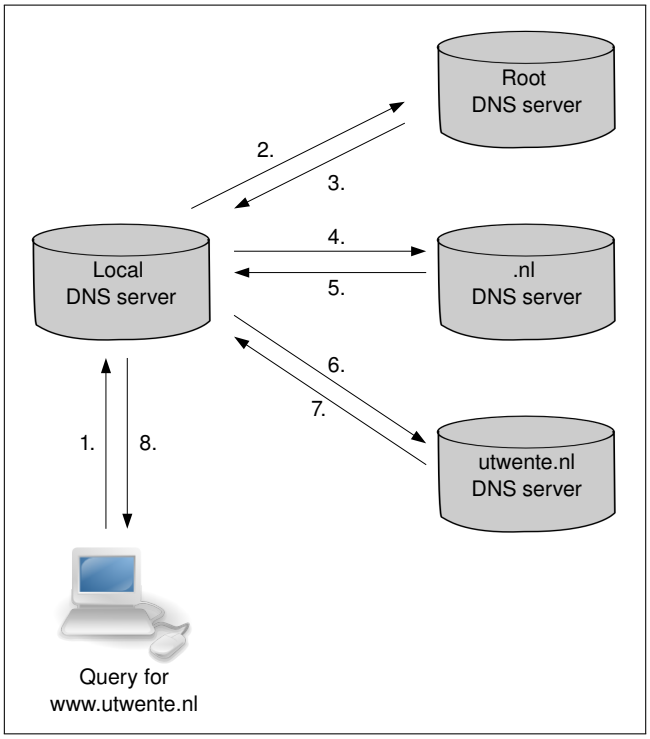


Figure 2.1: DNS message sequence: recursive querying

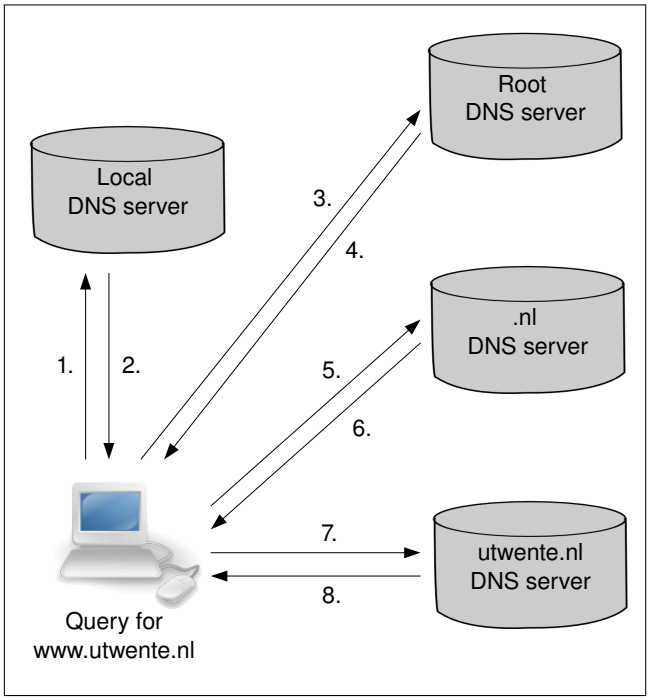


Figure 2.2: DNS message sequence: iterative querying

```
;; QUESTION SECTION:
;www.utwente.nl.                        IN      A

;; ANSWER SECTION:
www.utwente.nl.         86341   IN      CNAME   webhare.civ.utwente.nl.
webhare.civ.utwente.nl. 86400   IN      A       130.89.1.50

;; AUTHORITY SECTION:
civ.utwente.nl.         86400   IN      NS      ns1.utwente.nl.
civ.utwente.nl.         86400   IN      NS      ns2.utwente.nl.

;; ADDITIONAL SECTION:
ns1.utwente.nl.         86400   IN      A       130.89.1.2
ns2.utwente.nl.         86400   IN      A       130.89.1.3
```

Figure 2.3: DNS response for www.utwente.nl (using DIG)

If it has the address resolution in its cache it will return this, otherwise it will point to the DNS server with the best address fix.

The DNS protocol has record types. These record types can be found in the request and responses messages in the DNS protocol. In figure 2.3 a DNS response message to a query for address*www.utwente.nl* can be seen. Looking at figure 2.3 from top to bottom the following element can be seen:

- **Question section**: This section contains the initial address that was asked for.

  The **A** record indicates an address record, because we want the address for www.utwente.nl.

- **Answer section**: This section contains the IP address for the address we searched formulated.

  The **CNAME** record gives address names that are aliases of the searched for address. These records are called canonical name records.

  The **A** record here indicates that the host *webhare.civ.utwente.nl* has the latter IP address.

- **Authoritative section**: This section lists the DNS servers that are authoritative for the domain in which the address resides.

  The **NS** record indicates that host following the NS tag in this row is a DNS server for the domain in the beginning of the row.

- **Additional section**: This section lists additional information about the domain, in this case the address resolution for the authoritative DNS servers. This section is not obligatory, so there will be responses that do not contain this section.

The DNS protocol can return in total thirty four different record types. For this assignment it is of no further interest what these record types are, except for the ones that might be indicative for an attack. These will be handled when needed [22].

Every DNS record has a Time To Live (TTL) field. A DNS server will throw away records in its cache if the TTL has expired. The TTL field was intended for domain administrators to be able to make changes in their domain. In more recent years domain administrators started using the TTL field for load balancing. This is done by setting the TTL value to a very low value which forces a DNS server to query the authoritative DNS server more often, which will then return a server IP address which equalizes to the load of the server.

In figure 2.4 a small network is depicted. This network contains a primary and secondary DNS server run by the Internet Service Provider (ISP). This ISP has a link to the Internet. The ISP has two end users, each end user gets one IP address from the ISP. End user 1 uses this IP address to connect a computer directly to the ISPs network. End user 2 however wants to

connect two computers to the ISP network and uses a router to do this. For end user 1 it can be expected that when it does a DNS query it will not choose a random port number with a value lower than 1024 as these are restricted port numbers [23]. For end user 2 the same traffic might be expected, however the router that end user 2 uses contain a DNS server. The DNS server in the router is used by the Local Area Network (LAN) to resolve hostnames with that LAN. It is also used to resolve queries for addresses on the Internet that are asked by the LAN hosts. Most manufacturers have implemented the routers DNS server to issue these queries from port 53, this is conform the RFC 1035 specification. The use of port 53 as a source port number a for queries by DNS server can be seen as a security problem. D.J. Bernstein proposed to also use a random source port number for inter DNS traffic [6] instead of source port 53.

One feature of the DNS protocol which is reasonably unique is the use of both UDP and TCP as a transport mechanism. The DNS protocol has some rules about when to use UDP and when to switch over to TCP. The main rule is that, if a client contacts a DNS server for a query and the answer to this query makes the UDP packet larger than 512 bytes, the DNS server will respond a message stating that the use of TCP is needed to get the answer. A client is allowed to use TCP for its query instead of UDP even without first contacting the DNS server over UDP [41][42].

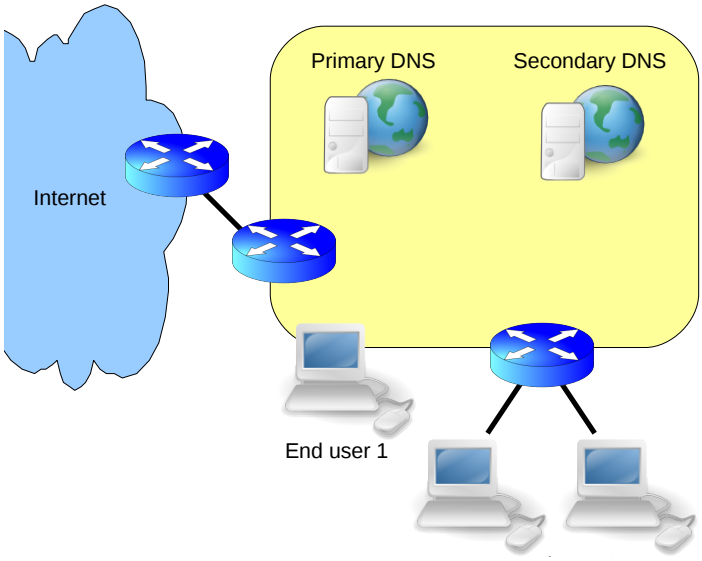


Figure 2.4: Small example network

2.1.1 DNS response codes

One of the header fields of the DNS protocol is the response code (RCODE) field. When the server responds to a query it will set the RCODE field to the appropriate value to indicate if anything has gone wrong, and if so what has gone wrong. The possible values that the RCODE field can have according to RFC1035 [42] are the following:

No error (0) If no error occurred during the processing the query the RCODE field will have value zero.

Format error (1) In the case that a query does not conform to the DNS specification, by setting incorrect values for certain field or leaving out parts, a format error response will be send.

Server failure (2) A server failure response is send if a query is conform the DNS specification but has fields set in such a way that the name server implementation cannot handle it. An example of this is sending two questions in one query, which is correct according to the specification but should not be used. A query like this would generate a Server failure response.

No such name (3) Indicates that the requested domain name does not exist. This error response is send by the authoritative DNS server for a zone in which the domain name could have existed.

Not Implemented (4) The request done by the client is correct but the function is not implemented.

Refused (5) The request done by the client is refused due to policy reasons. An example of this is the refusal to disclose zone information to outside clients.

Reserved (6-15) These values have been reserved for future.

2.1.2 Extensions to DNS

Over the years some extensions were developed for DNS, which are used to tackle some important issues that are present in the DNS protocol. In the following sections a few of these extensions will be described.

DNS zone transfer

DNS zone transfer is not a separate protocol from DNS, it a part of the DNS protocol which is also known by query opcode AXFR [41]. DNS zone transfer is used for database replication. The DNS zone transfer makes it so that a domain administrator only needs to change zone information of one DNS server. After which DNS zone transfer is used to transfer the zone information to the other authoritative DNS servers in the zone. Figure 2.5 illustrates a some example zones in the hierarchy of DNS, the zones contains all the information about the hosts present in the network.

In figure 2.6 a sample of zone information can be seen. In this example only one host exists, namely *router.openet*. The host has two canonical names which are also points of reference for the name server and the mail server in this zone as are given by the *NS* record and the *MX* record. The Start Of Authority (SOA) record indicates which hosts are Authoritative for this zone, in this example that would be *ns.openet.* and *router.openet.*

There are some issues with the use of the DNS zone transfer. The first issue is that of compatibility with the DNS software. Most servers have extra information that is stored in their database about the zone. This information can however not be configured using DNS zone transfer because then the software would not be compatible with other DNS software.

The second issue has to do with the size of zone information and its propagation to the other name servers in the network. It can be imagined that the zone information to be transferred can get very large when the network has a large number of hosts. It can also be imagined that some changes will be made in a zone over time. To overcome the problems with transferring large zone transfers during normal operation of a DNS server when changes are made an extension was created for DNS named incremental zone transfer (IXFR) [44] which works as follows: somewhere in time an IXFR command will be issued with a serial number set in the SOA information. This will result in the response of current SOA information of the server and the zone information. The serial number in the SOA is used as a version number for the last version of the zone that was seen. When a change is made to the network this serial number is changes.

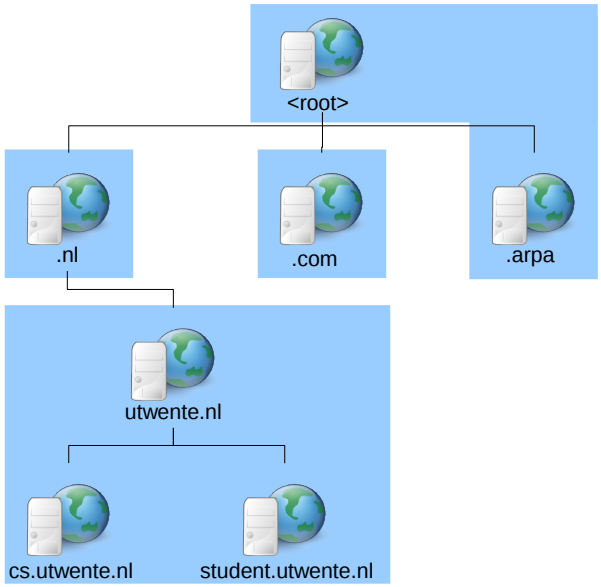


Figure 2.5: Zones in DNS hierarchy

openet.	86400	IN	SOA	ns.openet. router.openet. 2005062901 21600 3600 604800 86400
openet.	86400	IN	NS	ns.openet.
openet.	86400	IN	MX	10 mail.openet.
router.openet.	86400	IN	A	10.0.4.1
ns.openet.	86400	IN	CNAME	router.openet.
mail.openet.	86400	IN	CNAME	router.openet.
openet.	86400	IN	SOA	ns.openet. router.openet. 2005062901 21600 3600 604800 86400

Figure 2.6: AXFR zone information example

It can be incremented, but a best practice is to make it a time stamp. When a new IXFR command is issued with the old received serial number, the server will check if its version is the same or if it differs, and will send the changes since the last serial number.

Dynamic updates

By using the dynamic update protocol it is possible to update host information in a zone [60]. A dynamic update be used in a situation where hosts can become part of the network via a Virtual Private Network (VPN) connection. In order to make the hostname of the VPN host available to the DNS zone a dynamic update is performed by some authorized host, which can be the Dynamic Hosts Configuration Protocol (DHCP) server that issued IP address lease to the VPN host [2] [17]. Also when this user leafs the network it should be removed from the zone. The dynamic update protocol describes additional possible response code (RCODE) values to the list of RCODEs already described in section 2.1.1. The additions to the list are :

YXDOMAIN (6) The update request contains a name that should not exist.

YXRRSET (7) The update request contains a Resource record (RR) that should not exist.

Field Name	Field Type	Description
NAME	domain name	empty (root domain)
TYPE	u_int16_t	OPT
CLASS	u_int16_t	sender's UDP payload size
TTL	u_int32_t	extended RCODE and flags
RDLEN	u_int16_t	describes RDATA
RDATA	octet stream	attribute,value pairs

Table 2.1: Options Resource Record

NXRRSET (8) The update request misses a RR that is needed for the zone.

NOTAUTH (9) The DNS server is not authoritative for the zone which is declared in the update request.

NOTZONE (10) The update request uses a name in the Prerequisite or Update Section which is not within the zone.

Extension mechanism from DNS 0

This extension also called EDNS0 and can be found in RFC 2671 [58]. RFC 2671 focuses on three alterations on the DNS protocol two of which are alternate use of DNS header information. The third one is of interest to this project as it defines a mechanism to allow the transfer of packets larger than 512 Bytes over UDP. DNS clients and server that implement EDNS0 can include an Options Resource Record (OPT RR) with layout as in table 2.1. The class field in the OPT RR can be set to the value indicating the maximum size of the UDP packets.

Extension mechanism from DNS 1

This extension also called EDNS1 [59]. This extension allow a DNS client to ask multiple questions in the same query. Although this feature was already described in RFC 1035 it was flawed and never used [42]. The flaw that is present in RFC 1035 is that it is unclear how a server should react if one of the questions returns an error. EDNS1 was created to fix the problems that were not handled in RFC 1035. EDNS1 works on top of EDNS0, so a packet containing a EDNS1 request can be larger than 512 bytes. The use of ENDS1 allows a DNS client to significantly increase the size of a request by asking more than one question. Although this extension is still a draft it might have influence on the applicability of search method that will be discussed in this thesis.

DNS blacklists

A DNS blacklist (DNSBL) is a DNS database containing IP addresses of hosts that have performed suspicious activity like spamming [38]. A DNS blacklist can be queried in a normal way it can be asked for the A record or TXT record of a possible spammer. The DNS blacklist can than respond with either *no such name* indicating that this host is not in the blacklist or it can respond with an address. The returned address can be used to indicate what made the host look suspicious.

2.2 Network Intrusion Detection Systems

To explain what a Network Intrusion Detection System (NIDS) is it is first best to know what a Intrusion Detection System (IDS) is. An IDS system is used to detect attacks by analyzing incoming data on an end host. The IDS software uses fingerprints of attacks to detect if the it is under attack. These fingerprints contain information on an attack, like specific bits set in a TCP header or data content. The IDS software is there to protect an end host. Now what if there is an entire network to be defended? One option is to install IDS software on every host in the network, this however not very practical and it might be that certain attacks are not found. The other option is to use an NIDS system. In that case the entry points to the network from the Internet are equipped with an IDS system that scans all traffic that passes through. In that case all the traffic of an attack towards the network or from the network should pass through the NIDS system which can protect an entire network.

The problem that packet analysis performing NIDS systems are facing is the rapidly increasing bandwidth of networks. Processing a packet takes time and system resources. When faced with increasing amount of packets the total amount of used system resources and processing time will increase to a point where the NIDS system is unable to handle the sheer volume of data. For high volume networks a new type of NIDS system is needed, a system that uses less processing time and system resources.

Some methods of finding attacks in Netflow have already been researched. These methods will be described in the following sections.

2.2.1 Packet symmetry

The basic feature of any normal UDP DNS traffic is that the traffic is balanced. This means that for every request there should be one response. Because UDP is connectionless it is however possible that some packets are lost. This also means that no packets will be retransmitted. The use of packet symmetry using Netflow data has been a subject of research by Van der Sanden [33]. His research has proven that it is possible to use packet symmetry. An open issue that was described by Van der Sanden is the packet symmetry of TCP connections. With UDP the traffic should be fully symmetrical while for with TCP this is not the case. It was calculated that the balance between incoming and outgoing traffic might differ a factor two or three.

2.2.2 Cache poisoning detector

For a description of cache poisoning attacks see section 4.2 on page 31. Karasaridis *et al.* have described a method to detect cache poisoning attacks [29]. In the detection method request and response flows are counted between two destinations with the same bytes per packet (bpr). If the flow count of either direction surpasses a certain threshold, then an alert is given and using equation 2.1 a decision is made whether this traffic is a Cache poison attack or not.

$$bpr_{response} > bpr_{request} + bpr_{threshold} \tag{2.1}$$

2.2.3 Tunneling attack detector

For a description of DNS tunneling see section 4.3 on page 34. During an attack it is possible that more flows are received with a higher average packet size. Karasaridis *et al.* have created a system based on Relative entropy to calculate the difference between a normal packet size distribution and the packet size distribution during a time period [29].

2.3 The Netflow data

To be able to formulate search algorithms a fixed dataset is needed so that the outcome of search algorithm is repeatable. If the algorithms would be developed using only real-time data, then every time the algorithm would run another result can be expected. Real-time analysis is the ultimate goal of this project, but for formulating and testing queries a fixed dataset is needed. Some of the networks from which data was captured performed sampling where only 1 out of n samples was captured, this was done to decrease the load on the capturing devices. The datasets used in this project are from three sources, namely:

- **University of Twente**: Here traffic from the university campus is captured. The Netflow data from the University of Twente (UT) has no sampling.
- **Geant**: Geant is a European network, which is formed out a collaboration between twenty six research and education networks, the European Commission and DANTE, which covers thirty European countries. Because Geant is a multi gigabit data communications network the Netflow data from Geant was sampled at a rate of 1 out of 1000 samples.
- **Surfnet**: This is an Internet Service Provider (ISP) in the Netherlands focused at providing Internet for Educational institutes, and local governments. Surfnet has sampled the Netflow data at a rate of 1 out of 100 samples. This was done because of the high traffic load on the Surfnet routers.

These institutes captured the data at routers in their network. These routers were configured to export the Netflow data in a specific Netflow version, for Geant and Surfnet Netflow version 9 was used and the UT used Netflow version 5. The basic information that Netflow contains are:

Flow id A number by which a flow can be identified

Start time The time stamp indicating the start time of the flow

End time The time stamp indicating the end time of the flow

Source IP The IP address that was set in the source IP field in the IP header

Source port The port number from which the *source IP* was connecting

Destination IP The IP address that was set in the destination IP field in the IP header

Destination port The port number from which the *destination IP* was connecting

Packets The number of packets that make up this flow

Octets The total amount of octets that make up this flows

Protocol The number stating the protocol that was used in this flow. This can be 17 for UDP, 6 for TCP or any other value as given by RFC 790 [48].

TCP flags In the case that the used protocol is TCP this field can contain the numerical representation of the used TCP flags by the packets in this flow

Netflow contains more information which can be interesting in other settings but were not of interest for this project.

The Netflow data that was captured for the project was captured during one week august in 2007 and during one week of September in 2008. The UT also gave access to real-time Netflow data.

In figure 2.7 the layout can be seen of the UT, Surfnet and Geant. The Grey routers are the routers at which the Netflow data was captured. From the figure it can be seen that the UT has two paths to the Internet. One path goes via Surfnet and the other via ND-IX. The UT however also has a direct connection with the University of Delft and Eindhoven, which are not depicted in figure 2.7.

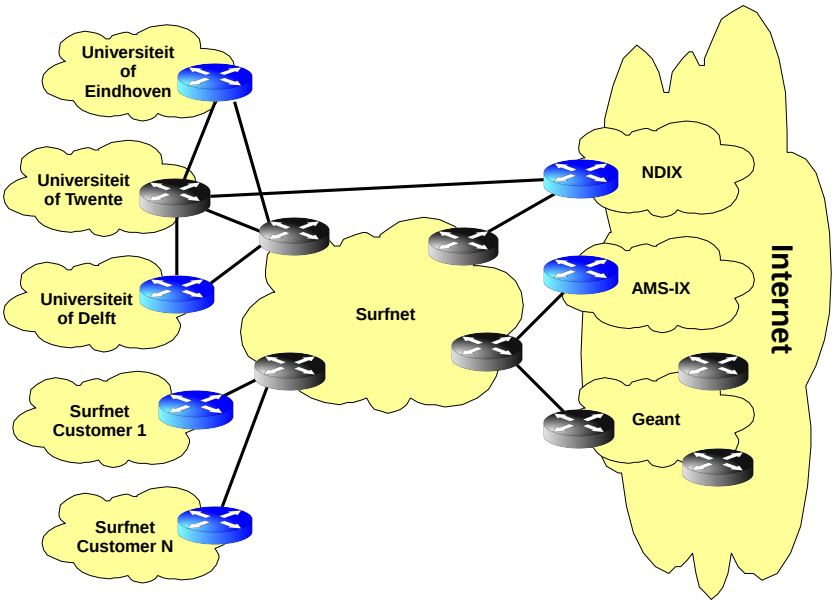


Figure 2.7: Network topology overview

2.4 The TCPdump data

TCPdump is a tool which can be used to capture traffic that arrives at a host or is sent by the host [40]. During the Netflow data capture in September 2008 the DNS traffic to and from the primary and secondary DNS server was captured. For both DNS servers the connecting switch was configured to mirror the traffic to and from the DNS server. This was done to separate DNS server operation from the traffic capturing. By using TCPdump to capture traffic it is possible to look into the content of the packets. In figure 2.8 a depiction is given of the mirror port setup. Traffic from the *DNS clients and DNS servers* (bottom of figure) destined for the *DNS server* is mirrored to the *mirror host*. Also the traffic from the *DNS server* to the *DNS clients and DNS servers* is mirrored to the *mirror host*.

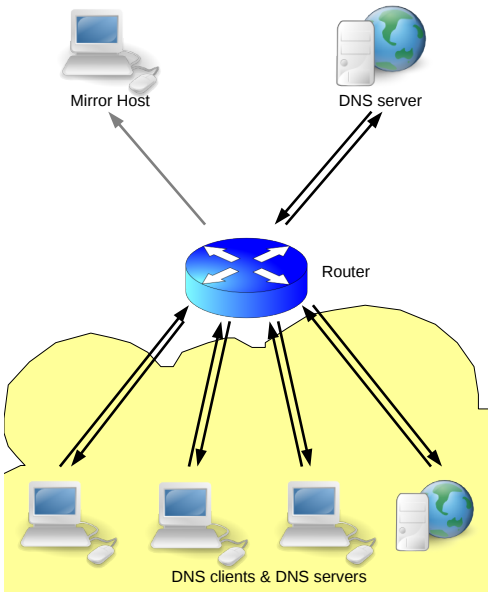


Figure 2.8: TCPdump capture mirror port setup

3 DNS statistics

In this chapter the statistics about the DNS servers and clients will be gathered from the Netflow data and the TCPdump data that was captured. By analyzing these statistics a more clear picture can be draw on the real life traffic of DNS clients and servers. In the first section the statistics that were gathered from the Netflow data will be given, and in the second section the statistics of the TCPdump data will be given.

3.1 Netflow data statistics

In this section the statistics that where gathered from the Netflow dataset that was captured in august 2007 will be discussed. Some questions were formulated that could be answered by the statistics. These questions are:

- How many DNS servers can been found in the Netflow datasets?
- Which part of the DNS traffic uses UDP and which part TCP?
- How much traffic is generated by these servers?
- How many clients connect to these servers?

In table 3.1 it can be seen that for the UT dataset only 0.91% out of all the hosts in the UT data is a DNS server. It is also visible that the UT dataset has a higher percentage of DNS servers compared to the number of active hosts. There are some possible reasons for this, namely:

- The UT Netflow data was not sampled, while the Geant and Surfnet data is sampled. This sampling allows for scarcely contacted host to not show up in the captured data.
- The UT Domain is a network with only end users. It can be assumed that some of these end users have routers installed with DNS software. As was described in section 2.1 these DNS servers contact other DNS servers via port 53.
- The high percentage of DNS server on the UT might be caused by scan attacks. As with scan attacks end hosts are contacted on port 53 while they are not a DNS server but do appear as such.

By measuring the number of port scans and the number of DNS servers within the UT it could be calculated which of these three factors has the biggest impact. However in section 2.3 it was already pointed out that not all traffic from the UT to the Internet passes through Surfnet. Another issue is the unknown impact of the sampling that was done on Surfnet.

	DNS servers	Active Hosts	% DNS servers
UT	221758	24284505	0.91%
Geant	94620	13913353	0.68%
Surfnet	196738	37655805	0.52%

Table 3.1: Balance of DNS servers against active hosts

	flows			packets		
	TCP	UDP	% TCP	TCP	UDP	% TCP
UT	102487	39165510	0.26%	452530	877979727	0.05%
Geant	21375	3067913	0.70%	25436	3350307	0.76%
Surfnet	62148	22983594	0.27%	88445	36791954	0.24%

Table 3.2: Balance of TCP versus UDP DNS traffic

From RFC1035 it is known that it is possible to connect to a DNS server using both TCP and UDP [42]. The basic mode of operation for the DNS protocol is run over UDP, and only in some situations the DNS protocol uses TCP. So it is interesting to know what percentage of traffic runs over UDP and how much traffic goes over TCP.

In table 3.2 a comparison on the amount of TCP traffic is made based on flow counts and the number of packets. The reason for this comparison is that flows display to number of time frames in which TCP or UDP traffic occurred, while the number of packets displays the amount of traffic that has occurred. One would expect that either way the percentage of TCP traffic would be almost the same. It can however be seen that in the UT domain the number of packets per UDP flow far exceeds the number of packets per flow of TCP. For all of the datasets it can be seen that the number of TCP packets is larger than the number of flows, which can be expected because of the three way handshake and connection teardown packets in TCP.

The growth of the amount of UDP is explainable by looking at how you surf the Internet. In most cases an end user will browse the Internet and almost always a website will be visited that has various hostnames that need to be resolved in order to display the entire webpage. Most web browsers are made such that they will resolve a list of hostnames using the same UDP socket and thus the same source port. One thing that can be clearly seen in table 3.2 is that only a very small percentage of the DNS traffic is caused by TCP.

Now that it is clear that in fact there is TCP traffic it is interesting to know how many DNS servers are only contacted using TCP or using both TCP and UDP. In table 3.3 it can be seen that there are some DNS servers that are only contacted via TCP, a reasonable assumption here could be that these DNS servers are contacted from other DNS servers. As stated in section 2.1 in most cases TCP will be used after a UDP address resolution has failed, so it can be expected that the amount of DNS servers with both incoming TCP and UDP connection is large.

	TOTAL	UDP	TCP	TCP+UDP
UT	221758	221672	86	0
Geant	94620	93051	571	998
Surfnet	196738	181541	13939	1258

Table 3.3: The number of UDP DNS servers of the total amount of DNS servers

	DNS packets	total packets	%
UT	920362476	32496256228	2.83 %
Geant	5379771	411652992	1.31 %
Surfnet	63317097	2286917973	2.77 %

Table 3.4: Amount of DNS traffic compared to total traffic

In table 3.4 the ratio between the total number of captured packets and the amount of captured DNS packets can be seen. For the UT and Surfnet an almost equal percentage of packets is for the DNS service. However for Geant this percentage is only half. The reason for this much lower amount of DNS packets is unclear. A reason might be that Geant in relation to the UT and Surfnet is a transient[1] network. The UT and Surfnet will have numerous end hosts contacting their local DNS server. For only a certain amount of queries does the local DNS server need to do recursive DNS lookups. The DNS traffic that can be seen on the Geant network will mostly by these recursive queries. So the lack of the queries to a local DNS server could explain this difference.

3.2 TCPdump traffic statistics

The TCPdump data that was captured at the primary and secondary UT DNS servers was analyzed for protocol usage statistics. Some papers on statistical analysis of DNS servers exists, some of these papers were used as a basis of identifying DNS anomalies. Wessels *et al.* [61] performed a TCPdump analysis on one root server. They identified eight different DNS query anomalies. Brownlee *et al.* [11] did similar measurement but had some added anomalies they searched for. Brandhorst *et al.* [9] analyzed TCPdump data of traffic captured at the UT router they searched for different DNS anomalies.

3.2.1 Measures

In this section the different anomalies that will be searched for in the TCPdump data will be described. The measures that were used by Wessels, Brownlee and Brandhorst will be described, some of which are not applicable on this TCPdump data.

Unused query class

In the DNS protocol the query class field can be set. This field is 16 bits in size but only 5 values are defined to be used, namely: IN (1), CS (2), CHAOS (3), HS (4) and ANY (255) [42]. It can be assumed that any other value for the query field is strange.

A record of an IP address

The response to an A record for a hostname will be an IP if it exists. There might however be DNS clients that ask for the A record of an IP address. This is of course a question that need not be asked by the client at all, nor should the question be recursively answered by the DNS server.

[1] A transient network is a network that consist mostly of routers and has hardly any end hosts

Unknown Top Level Domain

The Top Level Domains (TLD) can be separated into two groups, namely the country TLDs (.nl, .uk, etc.) and the generic TLDs (.com, .org, etc.). Currently there are 280 TLDs registered at Internet Assigned Numbers Authority (IANA) [24], any other asked for TLD should be considered as an Unknown TLD.

Non printable characters in query name

In RFC1035 it is stated that a queried name may only consist of the letters A to Z (capitalized or in lower case), numbers 0 to 9, and hyphen (-) [42]. In recent years this restriction has been lifted by extensions to the DNS protocol [31], therefore this cannot be seen as an anomaly anymore.

RFC1918

When a client queries a DNS server for a PTR record it is asking for a reverse DNS resolve. So a client wants a translation for an IP address to a hostname. In RFC1918 the network address space for intranet is specified. These addresses should not leave the intranet in which they are used. If a PTR record request is done for one of these domains then this was probably asked by a wrongly configured DNS client. The domains that RFC1918 state are:

- 10.0.0.0 - 10.255.255.255 (10/8 prefix)
- 172.16.0.0 - 172.31.255.255 (172.16/12 prefix)
- 192.168.0.0 - 192.168.255.255 (192.168/16 prefix)

Another element that was not mentioned by Wessels *et al.* [61] is the leakage of RFC 1918 addresses as a source address, although this has nothing to do with DNS it could contribute to the amount of repeated queries and the PTR queries done for RFC 1918 addresses.

Identical queries

If a client asks a DNS server for the same information with the same parameters (ID, type, class, name), then these queries can be classified as identical queries. There are two reasons why identical queries are strange phenomenon's, namely:

1. Asking the same question twice indicates a lack of response caching
2. By reusing the same query id the client is not implemented according to RFC 1035

Repeated queries

A repeated query is almost like an identical query, with the difference that the DNS client uses a different ID for every request. This gives information about the caching performed by a DNS client, because in case of repeated queries it can be assumed that this client does not cache responses at all or just for a very short period.

Referral not cached

A DNS client receives a referral when it asks a DNS server for a domain name that is part of its zone, but for which another DNS server is authoritative. For example a client gets a referral from the root DNS server to the .nl authoritative name server when it asks the root DNS server for www.utwente.nl. Now the DNS client should cache this referral. If the referral is not cached it will again query the root DNS server for a host within the .nl domain (for example www.google.nl). The problem with this query anomaly for the UT DNS servers is that a client that connect to the UT DNS servers mostly let the UT DNS server recursively resolve any query, so all the referrals are handled by the UT DNS server and not the clients. For this reason this type of anomaly was not searched for.

Error responses

As described in section 2.1.1 the response to a query will have an RCODE value. If this RCODE value is higher than zero, then the response is an error response. Error responses can be expected in any network, the amount of error responses and the reason for the error responses are of interest.

Restricted source port number

According to RFC 1700 [50] the port numbers in the range 0-1023 are considered restricted port numbers and should only be used by the registered services for those port numbers. So in the analysis the DNS clients should not be using a port number lower than 1024 to connect to a DNS server, with one exception being port 53.

Dynamic updates

Domain administrators have the option to update zone information at run time using dynamic updates [60]. Because the dynamic updates are only allowed from authorized hosts it would be strange to see other hosts trying to do dynamic updates. Any response value other than zero (NOERROR) is an indication of either a misconfiguration or an unauthorized attempt to update the zone information.

Query IDs

In every DNS packet a query ID needs to be set. In the first DNS clients and servers it was common practice to increment the ID by one every time a query was done. It has been realized that this is not a good approach as an attacker can guess the query ID very fast. The use of a Pseudo Random Number Generator (PRNG) should yield an unpredictable next query ID. However of many PRNGs that are used for different purposes including DNS server it is known that they are not random enough. A good PRNG should generate uniformly distributed values: from this it is expected that the query IDs will be equally distributed and any peaks could indicate a bad PRNG or no PRNG at all [54]. Another option to determine the randomness of the PRNG is to use Phase Space Analysis as described by Zalewski *et al.* [63].

3.2.2 Outcome

In this section the outcome of the described measures from section 3.2.1 will be given and discussed. The statistics were calculated over DNS traffic that was captured on the primary and secondary DNS of the UT using TCPdump. For the statistical analysis only the traffic over UDP was checked, because as discussed in section 3.1 only a very small amount of traffic is TCP.

	Primary	Secondary	All
General			
Packets (in)	297311723	37188564	334500287
Packets (out)	330257650	43789811	374047461
Queries	297061020	37152322	334213342
Responses	277233055	32547014	309780069
Query content[a]			
Recursion flag	290271971 (97.7%)	30183294 (81.2%)	320455265 (95.9%)
Iterative	6789049 (2.3%)	6969028 (18.8%)	13758077 (4.1%)
Dynamic update	166194 (0.06%)	7297 (0.02%)	173491 (0.05%)
Repeated	244084071 (82.2%)	25184308 (67.8%)	269268379 (80.6%)
Identical	87885926 (29.6%)	12298620 (33.1%)	100184546 (30.0%)
RFC1918	7645 (0.003%)	10149 (0.03%)	17794 (0.005%)
RFC1918 PTR	1292212 (0.43%)	37583 (0.10%)	1329795 (0.40%)
A of IP	502230 (0.17%)	121278 (0.33%)	623508 (0.19%)
Query responses[b]			
No error	222082290 (80.1%)	22267032 (68.4%)	244349322 (78.9%)
Format error	175 (0.00006%)	145 (0.0004%)	320 (0.0001%)
Server failure	2648313 (0.96%)	1515671 (4.7%)	4163984 (1.3%)
No such name	51996185 (18.76%)	8677624 (26.7%)	60673809 (19.6%)
Not implemented	546 (0.0002%)	543 (0.002%)	1089 (0.0004%)
Refused	135853 (0.05%)	13398 (0.04%)	149251 (0.05%)
YXDOMAIN	72 (0.00003%)	85 (0.0003%)	157 (0.00005%)
YXRRSET	1724 (0.0006%)	147 (0.0005%)	1871 (0.0006%)
NXRRSET	19932 (0.007%)	376 (0.001%)	20308 (0.007%)
NOTAUTH	0	0	0
NOTZONE	0	28 (0.00009%)	28 (0.000009%)

[a]Percentages of analyzable queries.
[b]Percentages of analyzable responses.

Table 3.5: Client to DNS server statistics

The TCPdump analysis was separated into two domains, namely: the traffic from the clients to the UT DNS servers and the traffic from the UT DNS servers to other DNS servers. This was done because the UT DNS servers can do recursive querying and this can give interesting information about the DNS traffic.

In the analysis the traffic between port 53 and port 53 was not split according to the initiator of the conversation as this would complicate the gathering of statistics. This has the effect that some traffic is seen as queries while in fact they are responses.

Unused query class

Because of problems with the output format of TCPdump it was impossible to create a perfect working analyzing tool for finding unused query classes. During the analysis of unused query classes it was found that no extraordinary query classes were used. The analyzing tool did output some entries that it found to be unused query classes, but were in fact caused by machines performing queries with space in the domain name.

	Primary	Secondary	All
General			
Packets (in)	54189989	13340421	67530410
Packets (out)	62204411	15609917	77814328
Queries	54056327	13301903	67358230
Responses	49247103	12018641	61265744
Query content[a]			
Repeated	25022296 (46.3%)	7136437 (53.6%)	32158733 (47.7%)
Identical	17138 (0.03%)	17907 (0.13%)	35045 (0.05%)
A of IP	130736 (0.24%)	33378 (0.25%)	164114 (0.24%)
Query responses[b]			
No error	37049727 (75.2%)	8314042 (69.2%)	45363769 (74.0%)
Format error	297829 (0.6%)	79325 (0.66%)	377154 (0.62%)
Server failure	1025951 (2.1%)	653194 (5.43%)	1679145 (2.74%)
No such name	10488637 (21.3%)	2771370 (23.06%)	13260007 (21.6%)
Not implemented	1647 (0.003%)	102 (0.0008%)	1749 (0.003%)
Refused	383312 (0.78%)	200608 (1.67%)	583920 (0.95%)

[a]Percentages of analyzable queries.
[b]Percentages of analyzable responses.

Table 3.6: DNS server recursion statistics

A record of an IP address

In table 3.5 it can be seen that 0.33% of the queries done on the secondary DNS server was for the address resolution of an IP address. The output of the log files also showed the request for the IP addresses occurred at a stable amount of the entire week. In table 3.6 it can be seen that of these request the DNS server tried to recursively resolve 33378 which is 27.5% of the 121278 as stated in table 3.5. It can be seen that these numbers are lower for the primary DNS server where only 0.17% of the requests for A records of an IP address. Of these requests the primary DNS server tried to recursively resolve 26.03%. The strange part in these recursively resolved queries is that, the DNS server should not even attempt to perform these requests.

Unknown Top Level Domain

In tables 3.7(a) and 3.7(b) the top ten requested TLDs can be seen all of which are not unknown TLDs. There are however a lot of unknown TLDs that where requested the first of which is on the 31 in tables 3.8(a) and 3.8(b) the top ten most requested unknown TLDs are given. It can be seen that the percentage of times that these unknown TLDs are requested in comparison to the total amount of request done by DNS client to the DNS servers is very small. What is interesting to notice is most of these TLDs (local, lan, mshome, lokaal) are intranet domains which might indicate misconfiguration in clients.

RFC1918

The use of RFC1918 IP addresses should not occur in the UT network, however the statistics show that for the primary DNS server forty different RFC1918 IP addresses and twenty-four for the secondary were used. For both the primary and secondary DNS server a significant part

(a) Primary DNS server

Domain	Amount	% of queries
arpa.	136086248	45.8%
nl.	92450838	31.1%
com.	32003696	10.8%
org.	12629577	4.3%
net.	12034560	4.1%
de.	1397324	0.5%
ru.	1039119	0.3%
uk.	518342	0.2%
dk.	447350	0.2%
edu.	364042	0.1%

(b) Secondary DNS server

Domain	Amount	% of queries
nl.	16979042	45.7%
arpa.	6288185	16.9%
com.	5812015	15.6%
org.	2750672	7.4%
net.	2547030	6.9%
ru.	263608	0.7%
de.	251251	0.7%
dk.	160783	0.4%
info.	129839	0.3%
uk.	116850	0.3%

Table 3.7: Top 10: Requested top level domains

(a) Primary DNS server

Domain	Amount	% of queries
local	246872	0.08%
wpad	43915	0.01%
icehp4	40225	0.01%
lan	27120	0.009%
manticore	24924	0.008%
lokaal	22918	0.008%
mshome	22749	0.008%
nld	22347	0.008%
id	19949	0.007%
melita	19914	0.007%

(b) Secondary DNS server

Domain	Amount	of queries
mshome	21455	0.06%
local	17485	0.05%
wpad	13703	0.04%
gs1	8321	0.02%
localhost	6563	0.02%
lan	6234	0.02%
lokaal	4535	0.01%
nld	4407	0.01%
belkin	2888	0.008%

Table 3.8: Top 10: Requested unknown TLDs

(25% primary, 42% secondary) of the IP addresses were used more than a hundred times in a request, some even up to two thousand or more.

On the DNS servers also a lot of PTR requests where done. For the primary DNS server a total of 1292212 requests where done, and for the secondary DNS server 37583. In percentages this means that

Repeated queries

In table 3.5 it can be seen that a high percentage of the queries done on the DNS servers is a repeated query. The most probable reason that this percentage is, that most traffic to the DNS servers is from end hosts. For these end hosts the UT DNS servers are the DNS cache. For the recursion rate this means that only a small part of requests are recursively resolved (18% primary, 36% secondary).

In table 3.6 it can be seen that of all the queries still around 50% of all the queries is a repeated query. It could be expected that the UT DNS server cache enough data so that repeating a query would be almost unnecessary. There are possible reasons for these repeated queries, namely:

- **Error response**: When an error response is send as reaction to a recursively performed

query the DNS server might not cache this response. In that way another recursive query to be done for the same query.

- **Recursion**: When a DNS server performs a recursive query operation it will in some cases query different DNS servers with the same query.
- **Cache expiration**: As stated in section 2.1 on page 5 some Domains use the DNS system for load balancing, this is done by setting the expiration timer of the result set to a very low value. This will lead to an increase in the number of requests for the same name.

It is unclear which of these possibilities has the biggest impact.

Identical queries

It should be expected that clients will not use identical query IDs let alone totally identical queries. However in table 3.5 it can be seen that about 30% of all the queries done is identical. This means that of around 40% of all the repeated queries is in fact an identical query (36% primary, 49% secondary). In table 3.6 it can be seen that only a small amount of the recursively performed queries is an identical query. Although this amount is low it is still strange to so this amount as it should be expected that the UT DNS servers have a good PRNG and perform no identical queries at all. One reason for these indentical queries by DNS servers is that these queries are actually responses to identical queries from DNS client.

Error responses

In tables 3.5 and 3.6 it can be seen that most error types happen only scarcely. The two errors that stand out are the *server failure* and *no such name*. The *no such name* error with its 20% overall is the largest contributor to error responses. From the log files it has become clear that most of these *no such name* errors can be attributed to DNS blacklist querying. Of the *server failure* errors 58% was caused by reverse DNS lookups. The other *server failure* errors were caused by DNS queries to a lot of different domains.

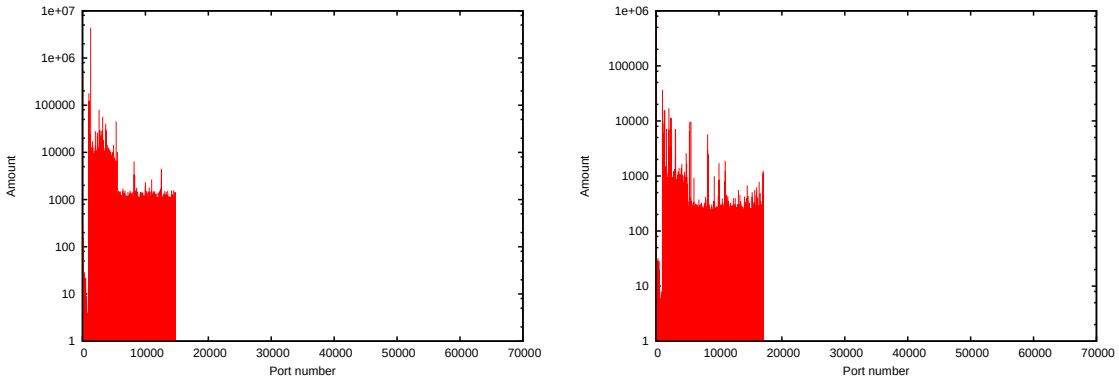
Restricted source port number

In figures 3.1(a) and 3.1(b) the distributions of the used source port numbers can be seen as used by DNS clients. It can be seen that the usage of the source ports is equally distributed. This means that there are DNS client that don't use a good random source port number. It can also be seen that there are request done using low source port numbers. In figures 3.2(a) and 3.2(b) the usage of the restricted port numbers is show in more detail. The peak for port 53 is because of DNS server that use source port 53 to perform recursive queries. But there are also normal clients that use source port 53 to contact the DNS server.

In figures 3.3(a) and 3.3(b) the distribution can be seen of the UT DNS servers recursive querying. It can be seen that both DNS servers have a nicely balanced use of source port numbers and do not appear to use restricted port numbers. The peak for port 53 in figures 3.3(a) and 3.3(b) is caused by response traffic from client connecting from source port 53. For these situations there is traffic from port 53 to port 53.

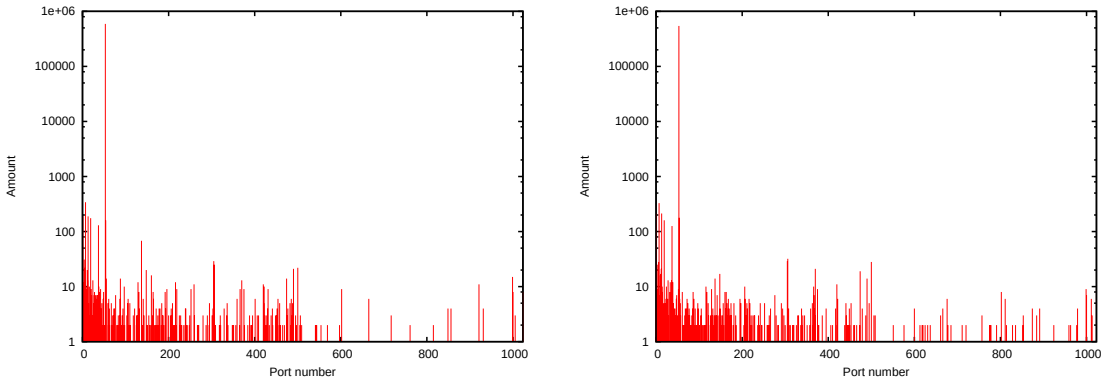
Dynamic updates

In table 3.5 it can be seen that from the requests to the UT DNS servers only a very small part was a Dynamic update query. Almost all of these requests where responded to with an error response. Most of the error responses that where sent in reaction to a Dynamic update query where *refused* errors messages.



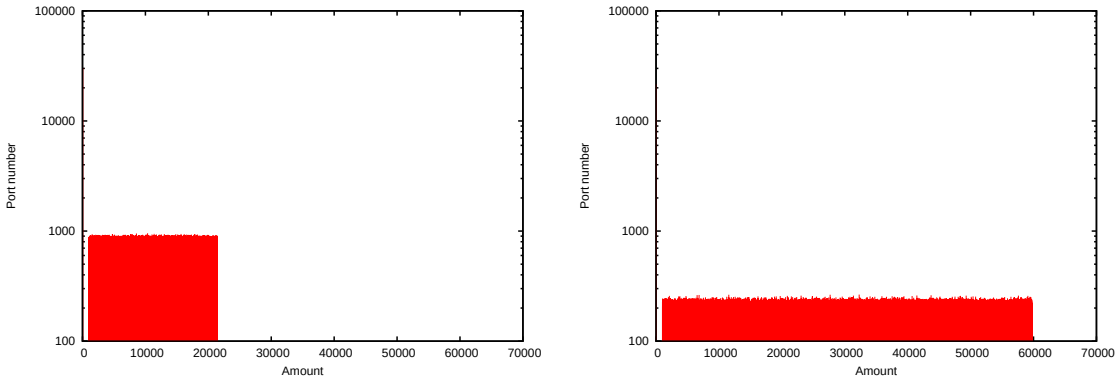
(a) Primary DNS server (b) Secondary DNS server

Figure 3.1: Source port distribution DNS clients



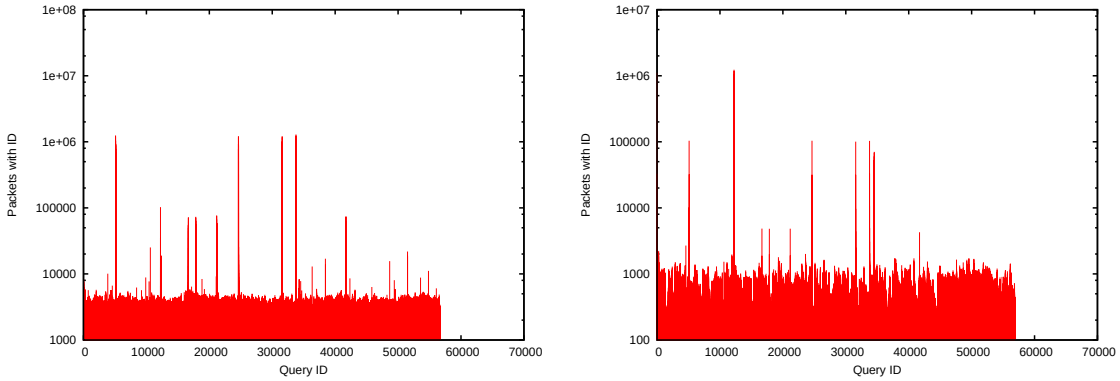
(a) Primary DNS server (b) Secondary DNS server

Figure 3.2: Source port usage Restricted range DNS clients



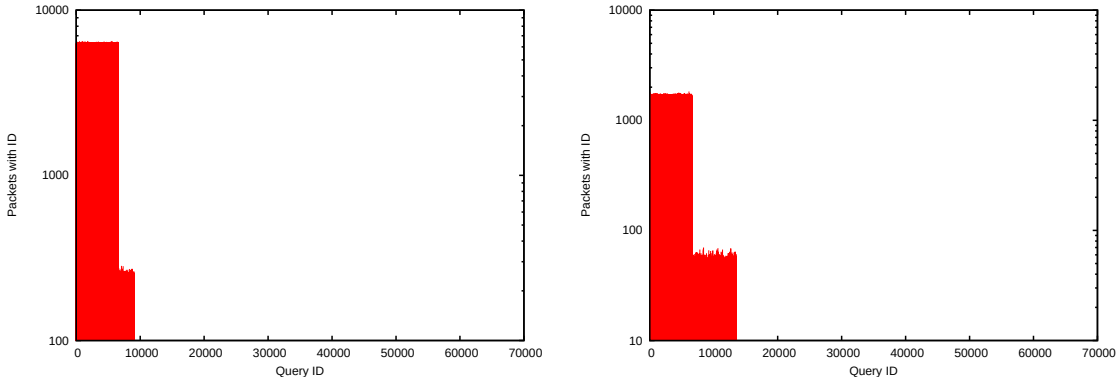
(a) Primary DNS server (b) Secondary DNS server

Figure 3.3: Source port distribution UT DNS servers



(a) Primary DNS server (b) Secondary DNS server

Figure 3.4: Query ID distribution DNS client



(a) Primary DNS server (b) Secondary DNS server

Figure 3.5: Query ID distribution UT DNS servers

Query IDs

In figures 3.4(a) and 3.4(b) the query ID distribution can be seen for the DNS traffic towards the DNS servers. In both figures it is clearly visible that there are large peaks for certain query IDs, this is a indication for the existence of badly implemented DNS clients that either use a fixed query id or have a poor PRNG. In figure 3.4(b) it can also be seen that the usage of specific query IDs fluctuates a lot. This fluctuation is a clear indication that some DNS clients use poor PRNGs, as with the use of good PRNG an almost uniformly distributed figure should be visible.

In figures 3.5(a) and 3.5(b) the query ID distribution can be seen of the DNS traffic generated by the UT DNS servers. It can be seen the UT DNS servers also do not have a uniformly distributed query ID usage. It can however be seen that apart from the increased usage of low query ID number the usage of most query IDs is evenly distributed. The peaks that appear with both DNS servers indicate an increased probability that one of these numbers is chosen. However if only looking at these numbers there is only a chance of approximately 0.015 percent that an attack can choose the right number, which is not very likely.

3.3 Summary

In this chapter it has been shown that only a small portion of the traffic on the Geant, Surfnet and UT network is caused by the DNS service. It has also been show that the UT DNS servers are primarily asked to answer a query in recursively, and that they only do this for approximately one sixth of the time. The statistics show that the DNS clients on the UT are not very smart client, this can be seen from the fact that approximately eighty procent of all the traffic are repeated queries or even identical queries. This can also be seen in the fact that address resolution is asked for IP addresses and the not uniformly distributed usage of source port or query IDs by the DNS clients.

4 Attack overview

Over the years attackers have found new ways to perform attacks to circumventing defenses laid down to protect against previous types of attacks. In this chapter the different types of attacks that involve DNS server will be described. The attacks involving DNS servers can be classified into two groups, namely:

- The attacks targeting DNS servers
- The attacks using DNS servers, to target some other system

In section 1 the basic types of Denial of Service (DoS) attacks will be described which can target DNS servers. After which a short description of attacks that target DNS server will be given in section 2, and in section 3 the attacks using DNS servers will be described. Finally in section 4 a detailed description will be given on identifiable traffic characteristics of the attacks as described in sections 2 and 3.

4.1 Types of DoS attacks

DoS are a known phenomenon on the Internet. It can be expected that DNS servers are also potential targets for a DoS attack. In this section the basics of a DoS attack as can be found in literature is described.

Before discussing the types of attacks let us first look at the flaw that is used in almost all DoS attacks. The flaw that is exploited is the absence of header security in IPv4, as can be seen in figure 4.1. The header of IPv4 holds the source and destination address of a packet. In a normal connection the source address of a packet would be set to your address, while the destination would be the address of the server you want to connect to [5]. By changing the source address to a different address than your own it is almost impossible to trace back where the packet came from or it can be used to attack a victim. In most networks nowadays network administrators are using ingress filter to disallow users within their domain to spoof their IP address [19]. Also egress filters are installed to disallow traffic for outside the domain to use IP addresses from within the domain [10].

Figure 4.2 depicts a DoS attack. From this figure the path from an attacker to the attacked host can be seen. Four actors can be distinguished in the figure, namely:

- **Attacker**: The attacker is the actor which initiates the attack. In many cases there is not just one attacker but multiple attackers which try to keep a host or network of the Internet for as long as possible.

0 0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
Version				IHL				TOS								Total length															
Identification																Flags				Fragment Offset											
TTL								Protocol								Header Checksum															
Source IP address																															
Destination IP address																															
Options and Padding																															

Figure 4.1: The IPv4 header

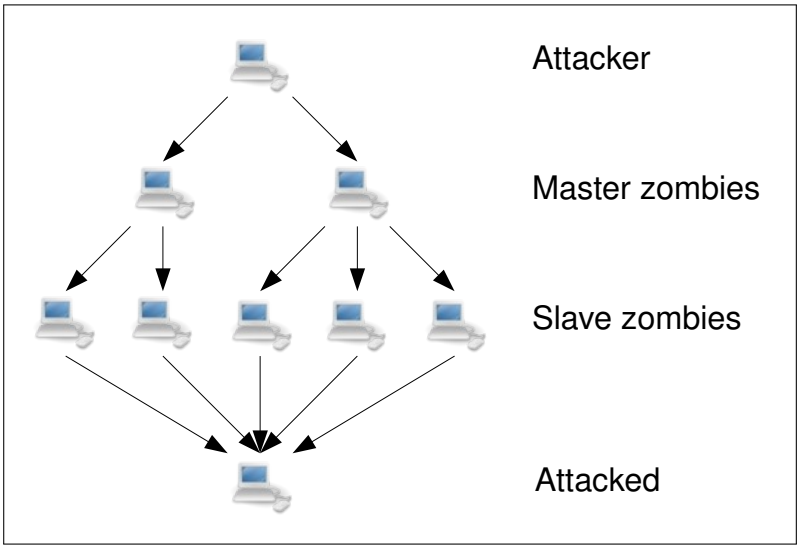


Figure 4.2: A DDoS attack using Botnets

- **Master zombie**: An attacker will in most cases use a number of botnets. The program used by the attacker to initiate an attack using a specific botnet is the Master zombie.
- **Slave zombie**: When a Master zombie is given a command to execute, it will pass this command to the Slave zombies it controls which will then execute the command.
- **Attacked**: The attacked is the host or network that is being targeted by the attacker.

These actors are common elements in DoS attacks. Attackers that perform a DoS attack are focused on disrupting a server by consuming resources of the server, which can be either the CPU, memory or bandwidth. The three types of DoS attacks that can be distinguish are:

- **DoS attacks**: This type of attack is the most basic. It is initiated by one or more hosts run by the attackers themselves. In most cases to make an attack of this type successful against a server, a lot of hosts need to participate, otherwise an attack will not take a large enough part of the bandwidth to take the server of the Internet. An attacker will in most cases exploit a flaw in the software of the server. That way a server can be forced to consume more resources with less effort [39]. Figure 4.3 shows what a DoS attack looks like. In *DoS attacks* the attacker will use a randomly generated IPv4 source address, if possible.

  An example of a DoS attack is the SYN flood, which uses a the TCP SYN packet to create half open TCP connections on the server, which lead to the server having a massive pool

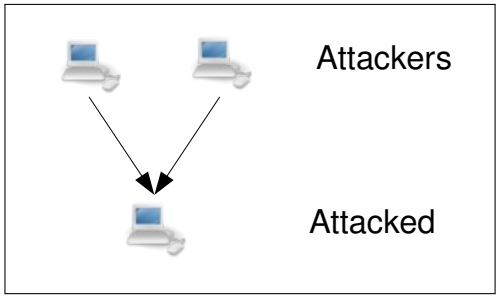


Figure 4.3: A DoS attack.

of half open TCP connections and not allowing for anymore connections from legitimate hosts. Other types of attacks use other TCP control packets like the TCP RST. Also ICMP and UDP packets are used, in certain attacks.

- **Botnet DDoS attacks**: A botnet is a network of hosts that are infected with a slave program, which accepts commands from the attacker. This type of attack was invented to tackle the problems encountered with *DoS attacks*. A trace back to the original attacker is in this case hard, because it only directs its slaves to attack, and does not participate in the attack itself. Only log files on infected host might allow for a trace back to the attacker [52][1].

  A slave (or bot) can be easily distributed over a lot of different hosts. This gives the attacker a large number of hosts to perform an attack with. This means that when he attacks, the bandwidth of the attacked server will more easily be flooded. Figure 4.2 shows what a *Botnet Attack* looks like. The attacker will have a certain amount of host called the Masters that are programs which maintain a list of Zombies. One of the ways a Master program maintains a list of active zombies is via an Internet Relay Chat (IRC) channels. The Zombie programs will after installation contact one or more Masters to notify that it is active [20].

- **Reflection DDoS attacks**: A *Reflection attack* is an attack that uses normal services running on servers across the Internet, to reflect innocent looking packets to the attacked host. In a *Reflection attack* an attacker will spoof this address and set it to the IP address of the host he wants to attack. By sending this spoofed packet to a service on the Internet a reply to this packet will be send to the attacked host instead of the attacker [28]. Figure 4.4 shows what a *Reflection attack* looks like.

Moore *et al.* did a backscatter analysis on DoS attacks by using the response from attacked servers [43]. This backscatter analysis resulted in the knowledge that around 90% of all the attacks use TCP followed by UDP on about 2% and ICMP with 2%. The analysis also showed that only about 0.5% of the attacks on the Internet used the DNS port. In the backscatter analysis they also found that 2-3% of the attacked hosts are DNS servers.

4.2 Attacks targeting DNS servers

In this section the different types of attacks that target DNS servers as can be found in literature are described. The attacks that can be found in literature are:

- **Basic DoS attack**: Because a DNS server is a host in the network it can also be attacked the same way as any other host on the Internet. However due to the high bandwidth

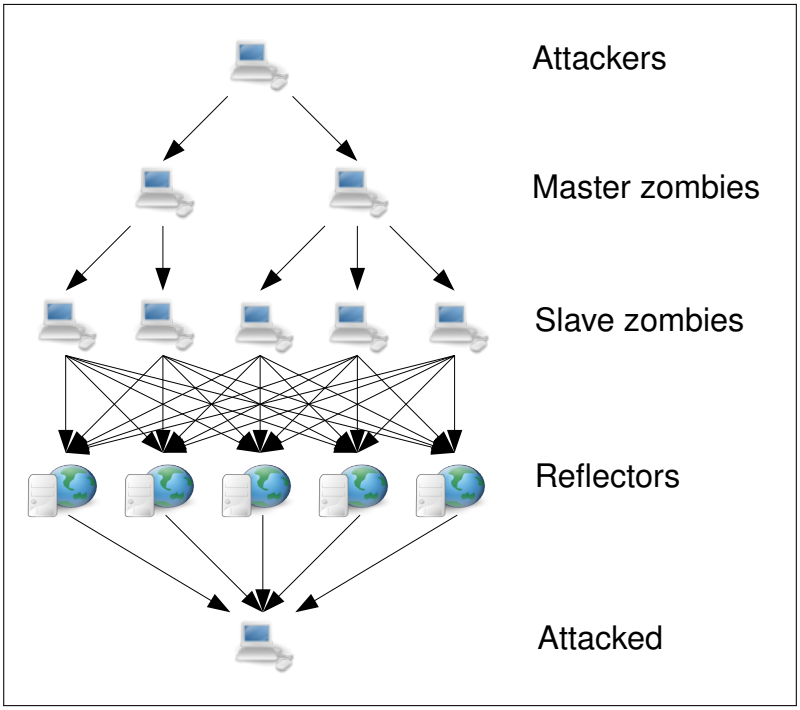


Figure 4.4: A Reflection DDoS attack

available to a DNS server a *DoS attack* will most likely have hardly any effect on the DNS servers performance, unless it is a small company DNS server or the DNS software contains flaws that can be exploited. It is far more likely that an attacker will use a *Botnet DDoS attack* or a *Reflection DDoS attack* to attack a DNS server.

- **Recursive query attack**: A special case of a *Reflection DDoS attack* on a DNS server is the *Recursive query attack*. The idea behind a *Recursive query attack* is to query a large number of DNS servers for an address that it does not know and thus forcing it to ask the attacked Authoritative name server. Because a DNS server maintains a local cache of DNS resolves, an attacker is forced to request a different site in every query. In order to use a DNS server for the *Recursive query attack* this server needs to have recursive querying enabled. An added feature in this attack is that if an attacker would spoof the from address and set it to be the attacked DNS server any reactions from the DNS servers will also be forwarded to the attacked DNS server [45].

- **DNS cache poisoning**: The absence of any signature validation on DNS entries makes it possible for an attacker to inject false resolves for a domain or host. Cache poisoning can be done on a DNS server with recursive querying enabled or an end host. The attack is initiated at the moment that a host asks for the address resolve. The cache poisoner will try to react to the query faster than the DNS server that is being queried, because then its answer to the query will be accepted. For this to work the attacker needs to know which DNS server is the Authoritative DNS server for this address resolution and then needs to spoof its source address to this DNS servers address. One approach to make sure that the Authoritative DNS server does not respond in time is to slow down the Authoritative DNS server by performing a DoS attack on it [51] [26].

  This type of attack is possible because the only protection against this is the query Id (QID) field in a DNS packet. In earlier times this QID would only be incremented by one

after every query, but it was soon realized that this would not provide any protection. All DNS servers used nowadays apply Pseudo Random Number Generation (PRNG) for the QID field. However this field is only 16 bit long and the quality of PRNG is in a lot of cases not as good as it needs to be [53] [63].

Recently a new *DNS cache poisoning attack* was found by Dan Kaminsky, making it possible for an attacker to perform the attack in as little as 10 seconds [8]. For this attack an attacker will setup a website on the Internet. This website will contain a large amount of links to a host within the domain they want to attack. These links will be given a hostname like AAAAA.VICTIM.COM to ZZZZZ.VICTIM.COM. In figure 4.5 the steps of this attack are depicted. When a client opens on of these websites it will try to resolve the addresses in these links. The Authoritative DNS server for VICTIM.COM will not know this host and return only it NXDOMAIN and not an A record. During this process the attacker tries to respond to requesting DNS server with a response saying that it knows the IP address of the requested hostname. It however also states a its DNS server as the authoritative DNS for the domain VICTIM.COM. Because the website will contain a lot of hostnames the chance that the attacker is able to be faster in his response than the Authoritative DNS server is very high, so eventually it will be able to poison the DNS cache for VICTIM.COM putting this domain under the control of the attacker.

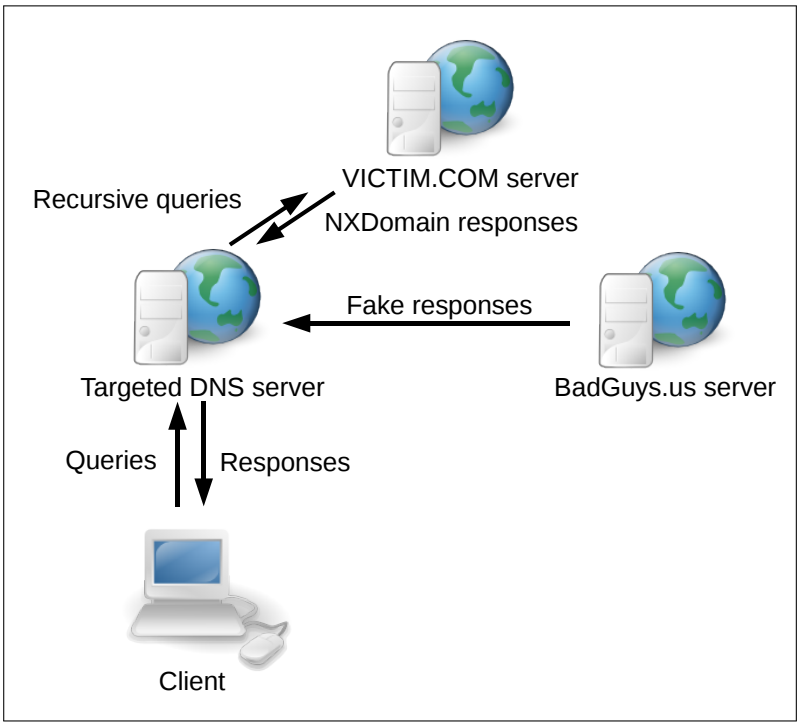


Figure 4.5: Cache poisoning — Dan Kaminsky

- **Buffer overflow**: As with a lot of software the existence of bugs in the DNS software make attacking some DNS servers with buffer overflow attacks possible [3]. A *Buffer overflow* is possible when software incorrectly handles the larger than possible content of a buffer which results in the overflow of the buffer into adjacent memory [32]. The bytes of data that overflow into the adjacent memory form the code that will be run. There are two ways in which an attacker can perform a buffer overflow on a DNS server, either by querying a DNS server or by being a responding Authoritative name server.

  One case is that the attacker performs a query on a DNS server. In this case the query

done by the attacker will have been manipulated so that the DNS server software causes a buffer overflow. This manipulation data itself is not detectable. The only thing that might indicate existence of a buffer overflow packet is size of the packet that is sent. The buffer overflow might already have succeeded after one packet, so no clear changes in packet size might be seen. One Computer Emergency Response Team (CERT) [13] advisory *CA-2001-02 Multiple Vulnerabilities in BIND* [36] shows multiple vulnerabilities that could be exploited in the DNS server BIND [55].

The other case is that the attacker uses an Authoritative name server. In this case the attacker manipulates the Resource Records it sends in response to a request, then the manipulated Resource Records causes the buffer overflow when it is read by the attacked DNS server. One such attack uses a flaw in BIND signature checking of Resource Records when they are received from another Authoritative DNS server [39].

- **Port scan**: A *Port scan* are used in an effort to discover services including DNS servers. A *Port scan* attack can do as little as just seeing if there is a reaction from a port, to as much as even trying some known exploits to gain access to the found service [37]. Most of these attacks are done by script kiddies that just download a port scan tool of the Internet. A port scan that provokes no reaction can be easily found, however if there is a reaction it is very hard to identify the traffic as a port scan.

4.3 Attacks using DNS servers

In this section an attack will be described that uses a DNS server.

- **Reflection attack**: As described earlier, an reflection attack uses a valid service to reflect is packets toward the intended attacked host. With DNS servers the use of *Reflection attacks* is interesting because a response packet from a DNS server is in most cases already three times larger than the original send packet. This amplification by using reflection attacks increases the effectiveness of a DoS attacks [21].

  An example of this amplification is: if an attacker would query for a resolution of *www.google.com* the DNS server will respond with a packet that is almost five times bigger. If the response of a large volume of queries like this is sent to the victim, the attacker will have reached its goal with five times fewer resources.

  An attack using the TCP is also possible. The feature of TCP that attackers use is the retransmission schema that makes a host retransmit packets a certain amount of times if the sent packages are not acknowledged by the other host.

- **DNS tunneling**: *DNS tunneling* is not considered to be an attack but it is a misuse of the DNS system. In *DNS tunneling* the content of the DNS packet is manipulated so that data (media, viruses) can be sent via DNS to a DNS server that acts as data server in the tunnel network. The manipulated parts of the DNS packet are the records. The query type can be any type, but certain values are more interesting like TXT record which allows BASE64 character encoding to be applied while an A record only allows BASE32 character encoding and no more than 255 byte of data [27].

4.4 Traffic characteristics of the attacks

Now that all the attacks involving DNS servers are known, it becomes of interest to look into the dynamics of the attacks. In this section the characteristics of the attacks described in section 2 and 3 are described.

4.4.1 DoS attack

In a DoS attack a single host will try to push a DNS server of the Internet. Although it is stated that it is a single host who should be attacking, it should consider that in most cases it is more likely that a small group of attackers is performing this attack, and the attack is performed directly from the attackers computer. The following characteristics can be seen in DoS attack:

- **A small number of hosts generate a large amount of requests**: However there are programs like spam filters that use DNS blacklists. This will also generate large amounts requests. This characteristic is only visible if the attacker has not spoofed the IPv4 source address.
- **Increase in the number of contacting hosts**: If the attacker has spoofed the IPv4 source address, then during an attack the number of hosts that contact the DNS server will be larger.
- **Reduced number of responses**: The idea behind a DoS attack is to overwhelm the attacked server with such a large amount of data that it cannot possibly handle every request, so that normal queries will not be handled. This should lead to the perception that there are more queries to the DNS server than there are responses from the DNS server. Analysis of this method of searching can be found in literature of Van der Sanden [56] and Kriebich*et al.* [33]
- **The average packet size might indicate a possible attack**: It can be calculated that an empty DNS packet will be 40 Bytes (8 Bytes for UDP header + 20 Bytes IPv4 header + 12 Bytes DNS header) [47] [41] [42]. This however is already strange because why send a message without a query. The smallest query that will not trigger a error response is a 45 bytes packet with 5 bytes of query data. During a DoS attack an attacker might use broken packets (44 bytes or less) or use other larger packets which either force the DNS server to perform resource intensive operations or occupies a large part of the available bandwidth.

  The assumption in this characteristic is that in a DoS attack either broken packets are send with only 43 Bytes or less, or that very large packets are send to create packet drops at the DNS server side.

4.4.2 Botnet DDoS attack

In a Botnet DDoS attack a large number of hosts will try to push a DNS server of the Internet. Here the characteristics of a attacking Botnet are described:

- **Increased amount of requests**: During a request the Botnet will be sending a massive amount of requests to the DNS server to overwhelm the available resources.
- **Reduced number of responses**: This characteristic has the same explanation as in section 4.4.1.
- **The average packet size might indicate a possible attack**: This characteristic has the same explanation as in section 4.4.1.
- **Increase in the number of contacting hosts**: Although a DNS server will be contacted a lot, an assumption can be made that when a DNS server is attacked by a Botnet there will be an increase in the number of contacting hosts.

- **A larger than average amount of client connect from areas with a high number of infected clients (Bad neighborhood)**: There are Botnets that can send both spam messages and be used for DDoS attacks [12]. To be able to use this characteristic some assumptions need to be made, namely:
  - There is a way to create a map indicating bad neighborhood. The bad neighborhoods should be based on the amount of hosts that have been compromised in a fixed IP range.
  - The Botnet hosts do not spoof the source address in the DNS query. If a Botnet does spoof its source address then the Bad neighborhoods holds no information.
  - A bad neighborhood will hold more than one type of Bot. If a neighborhood has several types of Bots it is more likely that there are both Spam Bots and DDoS Bots.

4.4.3 Reflection DDoS attack

For Reflection attacks the following characteristics can be found:

- **In a Reflection attack it would appear that a single host is asking for a lot of DNS resolves**: At first glance some would say this might look as a *DoS attack*, which is partially right. But because the end host is the target and not the DNS server, looking at the packet symmetry will most likely not show any anomalies. This statement has a counter example, namely: When the attackers launch their attack they will in most cases not be in the same network as the attacked host. The point of data capture will lead to the absence of the query to the DNS server and only the response will be present in the Netflow data, or vice verse.
- **An increased amount of contacted DNS servers by the attacked host**: In some cases more than one DNS might be used in this attack, which when looking at the data would be visible by the increase in the amount of contacted DNS servers.
- **Amplification**: One of the attractive reasons to use *Reflection attacks* is the amplification of octets in the response message in comparison to the request message. A large response packet to a query is very normal for DNS traffic, so detecting the use of amplification for an attack might be very hard.
- **Retransmissions**: With TCP based transmission a unacknowledged packet will have a retransmission. In case of a *Reflection attack* a packet will never be acknowledged and thus there will be retransmissions of the query response.

4.4.4 Recursive query attack

For a Recursive query attack the following characteristics can be found:

- **A high number of different queries will be done to a specific domain**: The specific domain cannot be seen in flow data because we do not know the content of the packets, the increased amount of requests could be detectable.
- **Increased amount of recursive queries to attacked DNS server**: In the outgoing traffic there will be a peak in the number of queries to a single DNS server from one or more DNS server during a certain period.
- **DoS on attacked DNS server**: At the side of the attacked DNS server the characteristics of any form of DoS attack should be visible.

4.4.5 DNS cache poisoning

From the basic definition of the DNS cache poisoning attack it is known that the attack has two parts, namely: the injection and the possible DoS on the Authoritative name server. This leads to the following characteristics for the DNS cache poisoning attack:

- **High number of responses to a query with the only difference being the query-ID field in the DNS header**: This large volume of responses is due to the query-ID guessing. From a large number of Pseudo Random Number Generators (PRNG) used in DNS software it is possible to guess within certain number of packets what the next query-ID that will be generated by the PRNG will be. The number of responses will differ depending on the DNS servers PRNG [53] [63]. The content of the response packets cannot be seen in Netflow, but the high number of responses can be seen.
- **Increased amount of Recursive DNS requests to a specific DNS server**: In a recently found attack possibility the attacker will request host resolutions for a large number of hosts within a domain [8].
- **Possible DoS attack on authoritative name server**: The DoS attack done on the Authoritative name server will have the same characteristics as a Basic DoS attack, it however does not tell anything else than that the Authoritative name server is under attack.

4.4.6 Buffer overflow

The following characteristics can be found for Buffer overflow attacks:

- **Malformed packets**: The content of a DNS packet will be manipulated in such a way that a buffer in the DNS server software will overflow.
- **Increased packet size**: To perform a packet overflow the manipulated data will need be large enough to make the buffer overflow and to execute a usable code.

4.4.7 Port scans

Although most Port scans are not only focused on the DNS port, there are instances were a host just wants to know if a certain host is running a DNS server. If a host that is actually running a DNS service is port scanned, the presence of a reaction from port 53 will make it hard if not impossible to detect a scan that is just focused on the DNS port. There are several ways an attacker can scan a host. These ways are depicted in figure 4.6 and are:

- **Horizontal scan**: a horizontal scan is a scan that is done from one host to many host within that same network on a service of a specific host.
- **Vertical scan**: a vertical scan is performed by connecting to a range of ports a single host, during a period of time.
- **Block scan**: a Block scan is a combination of a horizontal and a vertical scan.

Kinable did research into these different scan types and their detection possibilities using Netflow data [30]. Because of the focus on DNS servers the only interesting scan type is the horizontal scan, which has the following characteristics:

- **A host will make connections to a range of hosts in a specific domain**: Over a period of time an attacker will connect to several hosts within a domain.

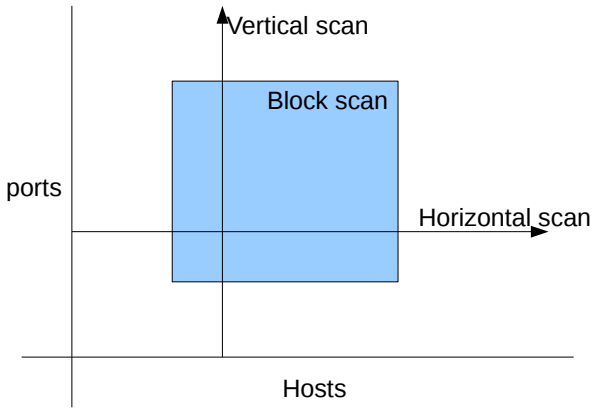


Figure 4.6: Port scan features

- **A host being scanned will not respond to a scan**: This characteristic has a side note, namely: If a host is running a firewall then the host will indeed not respond. If however the host is not running a firewall then with UDP it will respond with an ICMP packet and in case of TCP a TCP RST packet will be responded.

4.4.8 DNS tunneling

For a DNS tunneling the following characteristics can be found:

- **large amount of traffic**: A client connecting to a DNS tunneling server will generate a continuous stream of packets with queries for one specific domain. The flow will be continues for the DNS tunnel software that uses it to transfer large files or remote terminal sessions [46].
- **Mostly large packets will be sent over the DNS tunnel**: The use of DNS tunnels for file transfer makes the software stuff the packets to the maximum size allowed. If the file is large enough, multiple packets will be needed to send the data.
- **DNS tunneling uses some record types that are not used very often**: An example of this is the TXT record type, which is only used by spam DNS blacklists since a short time [38].
- The use of BASE64 and BASE32 character encoding makes the queries that are done look strange in comparison to regular queries.

5 Search methods for finding attacks

The characteristics of attacks as described in chapter 4 will have a specific way to be found in Netflow data or are impossible to find in Netflow data. In this chapter the methods for finding specific characteristics in an attack will be given, after which a few more general approaches will be described that could be used to find a set of attacks.

5.1 Search method per attack characteristic

The characteristics as described in chapter 4 can in most cases be seen using Netflow data. In this section the characteristics of each of the attacks will, if possible, be linked with a detection method.

5.1.1 DoS attack

- **A small number of hosts generates a large amount of requests**: This characteristic might be found by doing a time series search for increases in requests by a host.
- **Increase in the number of contacting hosts**: In a time series analysis an increase in the amount of contacting hosts might indicate a possible attack.
- **Reduced number of responses**: A time series analysis of the ratio between the number of request and the amount of responses could be used to find the reduced responsiveness of a DNS server.
- **Packet size anomalies**: A time series analysis of the change in distribution of average packet size could indicate the existence of packet size anomalies.

5.1.2 Botnet DDoS attack

- **Increased amount of requests**: This characteristic might be found by doing a time series search for increases in the amount of requests to the DNS server.
- **Reduced number of responses**: A time series analysis of the ratio between the number of request and the amount of responses could be used to find the reduced responsiveness of a DNS server.
- **Packet size anomalies**: A time series analysis of the change in distribution of average packet size could indicate the existence of packet size anomalies.

- **Increased amount of connecting hosts**: A time series analysis could show an increase in the amount of connecting hosts to a DNS server.
- **Hosts connecting from bad neighborhoods**: This characteristic might be found by using DNS blacklists to check if the connecting hosts are in a DNS blacklist. This is however a different use of DNS blacklists, which are normally used to indicate if a host, has sent spam email. It might however be that hosts that are used for spamming are also used for DDoS attacks.

5.1.3 Reflection DDoS attack

- **Apparent increase in requests by host**: This characteristic might be found by doing a time series search for increases in requests by a host.
- **Apparent increase in responses to the host**: This characteristic might be found by doing a time series search for increases responses to a host.
- **Increased amount of contacted DNS servers**: Doing a time series analysis of the amount of DNS servers that respond to a host might indicate a possible attack.
- **Amplification**: The packet size of the responses send to a host will be large, a time series analysis of the change in the distribution of the average packet size could show the existence of an attack.
- **Retransmissions**: A time series analysis of the ratio between the number of requests and the amount of responses could be used to find the increased responsiveness to a host.

5.1.4 Recursive query attack

- **Increased amount of requests by a number of hosts**: By doing a time series analysis it should be possible to see an increase in the amount of requests to a DNS server.
- **Increased amount of recursive queries**: By doing a time series analysis on the requests done by DNS servers, an increase in the amount of recursive queries to a specific host might be detectable.
- **Target DNS server under DoS attack**: This of course means that any of the detection methods for detecting *DoS attack*, *Botnet DDoS attack* or *Reflection DDoS attack* might be applicable for detecting an attack on the target DNS server.

5.1.5 DNS cache poisoning

- **High number of responses with only different query-ID**: Because no packet content is present when using Netflow the value of query-ID is unknown. A time series analysis of the ratio between the number of request and the amount of responses could however be used to find the increased responsiveness of a DNS server.
- **Increased amount of requests**: By doing a time series analysis it should be possible to see an increase in the amount of requests to a DNS server.
- **Increased amount of recursive queries**: By doing a time series analysis on the requests done by DNS servers an increase in the amount of recursive queries to a specific host might be detectable.

- **Authoritative DNS server under DoS attack**: This of course means that any of the detection methods for detecting *DoS attack*, *Botnet DDoS attack* or *Reflection DDoS attack* might be applicable for detecting an attack on the target DNS server.

5.1.6 Buffer overflow

- **Malformed packets**: The content of the packets in not present in Netflow.
- **Increased packet size**: A time series analysis of the change in the distribution of the average packet size could show the existence of an attack.

5.1.7 Port scans

- **Requests to a large number of hosts in domain**: Although the number of DNS servers a host will contact as normal traffic can be very high when performing integrative querying, a host should not only contact a limited amount of DNS servers per IP range.
- **Absence of response**: By checking if according to Netflow a request is done, but no response, ICMP or TCP RST packet is returned, it might be possible to detect a port scan.

5.1.8 DNS tunneling

- **Large amount of packets**: This characteristic might be found by doing a time series search for increases in the amount of requests to and responses from the DNS server.
- **Increased packet size**: The packet size of the request to a DNS server and the response from a DNS server will be large. A time series analysis of the change in the distribution of the average packet size could show the existence of an attack.
- **Hardly used record types**: This characteristic cannot be found using Netflow, because the packet content is not known.
- **Strange questions**: This characteristic cannot be found using Netflow, because the packet content is not known.

5.2 General detection methods

In the previous section a simplified view was given on methods to detect attack an characteristic. It could be seen that some of these methods have overlap, from this overlap the general detection methods that will be discussed in this section where formulated. This section might not show all the possible detection methods. The methods that are shown in this section might also not be directly applicable and are left as future work.

5.2.1 Packet per flow method

One thing that stands out for all of the attacks is the volume of traffic that needs to be transmitted to and from a DNS server. This amount may differ from a few thousand to infinity. The volume of traffic is comprised of the number of packet and their total size. From chapter 4 it is known that the packet size can indicate some types of attacks, however for most attacks the content of the packets is not an important factor. On the other hand, every packet a client sends indicates a query and it can be expected that a normal client will perform a small amount of

queries during a certain time period. A malicious client on the other hand will have flows with a large number of packets, or will generate a enormous amount of flows with only a small amount of packets.

The duration of flows in Netflow is limited by a parameter called *ip flow-cache timeout active <minutes>*, within this period the number of packets can take on any value [14]. This method focuses on the effect that a DNS server that is not under attack will have a stable overall amount of packets per flow. Only when an attack occurs does this amount of packets in a flow increase. A peak can however also be normal traffic. Some type of calculation needs to be found identify if the values of packets per flow that are seen contain anomalous behavior. This method could be used to detect the detect following attacks:

- DoS attack
- Botnet DDoS attack
- Reflection DDoS attack
- Recursive query attack
- DNS cache poisoning
- DNS tunneling

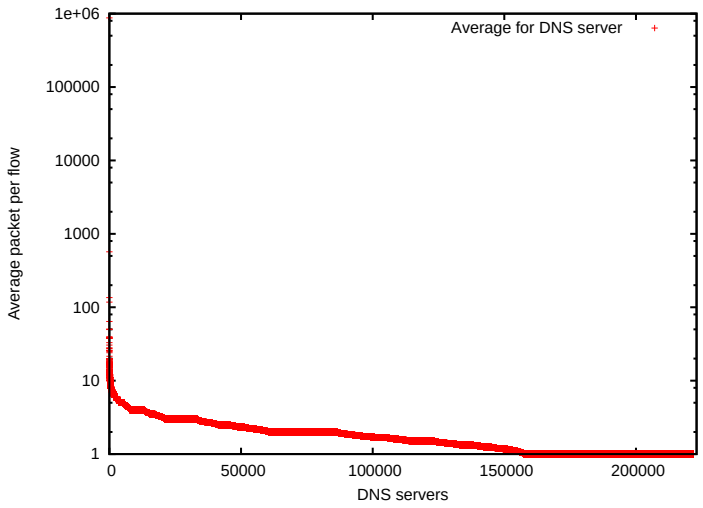


Figure 5.1: Distribution of average packets per flow — UT dataset.

Figure 5.1 shows a plot of the average number of packets per flow for every destination of DNS traffic for the UT Netflow dataset. From this figure it can be gathered that the largest amount of hosts has an average number of packets per flow lower than 10. Although this average does not say anything about the server being attacked or not, it does show that a large number of servers have an almost similar average. From this it can be argued that from a server with normal traffic this average will deviate only little over the duration of time.

Standard deviation

To make a graph for every single server with this average as function of time would be time consuming. To overcome this obstacle a solution might be to look at the deviation between the average number of packets per flow for a server over the entire period and the flows containing

the largest number of packets. In this the assumption is that the servers with a deviation greater than a certain threshold can be assumed to contain malicious traffic. The height of this threshold is unknown so an appropriate value for this threshold needs to be calculated.

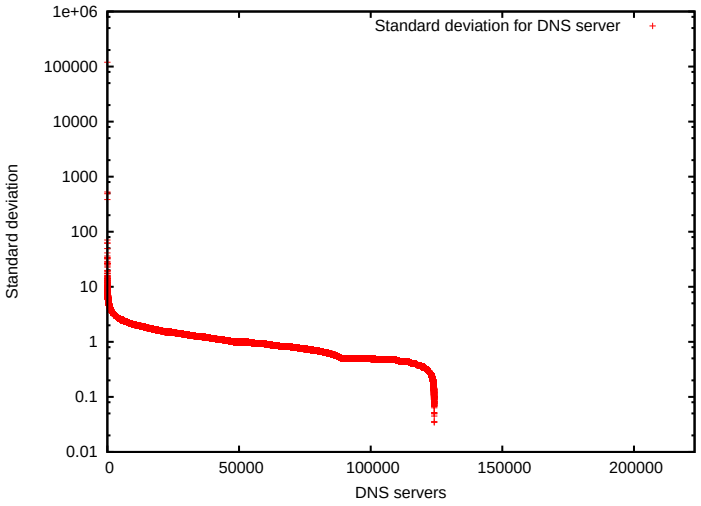


Figure 5.2: Standard deviation packets per flow — UT dataset.

In figure 5.2 the standard deviation curve for DNS servers in the UT dataset DNS can be seen. To calculate this standard deviation equation 5.1 was used. It is shown that a large number of DNS servers have a standard deviation of around one. The DNS servers that could have been attacked are assumed to be located on the far left side.

$$\sigma = \sqrt{\frac{1}{N}(\sum_{i=1}^{N} x_i^2 - N\bar{x}^2)} \tag{5.1}$$

There is a problem with using only standard deviation is that the value of the standard deviation, because increased by the values that are beneath the mean. In a Normal distribution it can be expected the sample values are distributed equally around the mean value. The measure to describe if this is actually the case is called the skewness. The skewness of a distribution can be calculated using formula 5.2. If the skewness is zero then the values are normally distributed, however if the skewness value is positive the distribution will have a long right tail with the mass of the distribution being on the left side and if the skewness is negative the distribution has a long left tail with the mass of the distribution being on the right side. In figure 5.3 a graphical representation of what skewness is shown.

$$g_1 = \frac{\frac{1}{N}(\sum_{i=1}^{N} x_i - \bar{x})^3}{(\frac{1}{N}(\sum_{i=1}^{N} x_i - \bar{x})^2)^{\frac{3}{2}}} \tag{5.2}$$

An analysis of the distribution of several known servers shows that these are exponentially distributed and not normally distributed. This makes using skewness an uninteresting approach. In appendix B some test results are shown on the use of the standard deviation method.

The idea of calculating the standard deviation of the entire period is not very practical for real life application. This is because in a real life application using a entire dataset for a calculation will cost more time than one analyzing the new flow information. It would be more realistic to calculate the standard deviation for time frame. This might be done by looking for increases in the standard deviation over time.

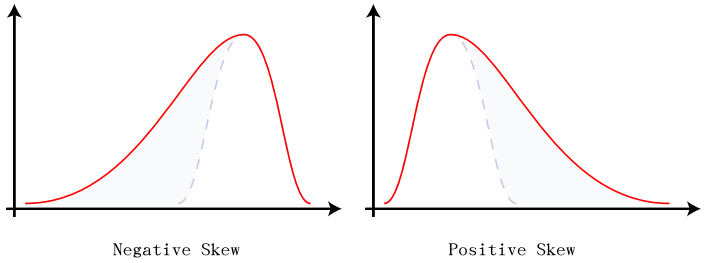


Figure 5.3: Skewness illustration (source Wikipedia)

Relative entropy

The definition of entropy is: "measure of disorder: a measure of the disorder that exists in a system" [18]. The measure of Relative entropy or Kullback-Leibler distance is the measure of difference between two probability distributions bases on the entropy of these distributions [35][16].

Relative entropy might be applicable to the Packet per flow distribution if the probability distribution of a DNS server is relatively stable, and the probability distribution of a attacked DNS server will be different. The two probability distributions that will be compared are:

- **Reference distribution**: the reference distribution is the packet per flow distribution for a time period that shows normal traffic for a particular DNS server. The reference distribution will be denoted as $G(i)$ in equations for Relative entropy.
- **Current distribution**: the current distribution is a distribution for a time period for which it is unknown if it contains anomalies. Only by comparing it to the reference distribution should it be possible to detect anomalies. The current distribution will be denoted as $F(i)$ in equations for Relative entropy.

The idea for this measure came from a paper by Karasaridis *et al.* who used Relative entropy to find DNS tunneling by looking at the packet size distribution [29]. The measure of Relative entropy can be given using formula 5.3.

$$D_{kl}(F \parallel G) = \sum_{i=1}^{N} F(i) \log_{10} \frac{F(i)}{G(i)} \tag{5.3}$$

$$D_{kl}(F \parallel G) = \sum_{i=1}^{N} F(i) \log_{10} G(i) - \sum_{i=1}^{N} F(i) \log_{10} F(i) \tag{5.4}$$

Formula 5.3 can be rewritten to formula 5.4. The first part of formula 5.4 is called the Cross entropy formula, and the second part is called Self entropy. In the case that the Cross entropy and the Self entropy are equal, then the two distributions are equal and the Relative entropy becomes zero. The Relative entropy value will only increase if the difference between the two distributions increases. The functions *F(i)* and *G(i)* give the probability that this distribution has flows with i number of packets. It should never happen that the Relative entropy becomes negative.

For every DNS server a distribution of the number of packets per flow can be made and used to calculate Relative entropy values. However there are some side notes that need to be made, namely:

- If the number of flows is not high enough, the Relative entropy value might fluctuate very much. This has to do with the incompleteness of the probability distributions used for the calculations. This problem could also occur in periods in time when a DNS server has less traffic.
- If only a small part of some DNS server traffic was captured an incomplete probability distribution might result in more unstable probability distributions and thus a higher Relative entropy value, which might fluctuate more.

An example of the first side node is the following: During one period only flows with i number of packets are received. This would result in a probability of one for function *F(i)*. This makes the Self entropy of the system zero. If we would calculate the Cross entropy with an complete probability distribution the value of the Cross entropy can by any value and will be larger when the probability of *G(i)* decreases. Most importantly the Cross entropy will be the only factor for the value of the Relative entropy.

In figure 5.4 two identical probability distributions are shown, which has a Relative entropy of zero.

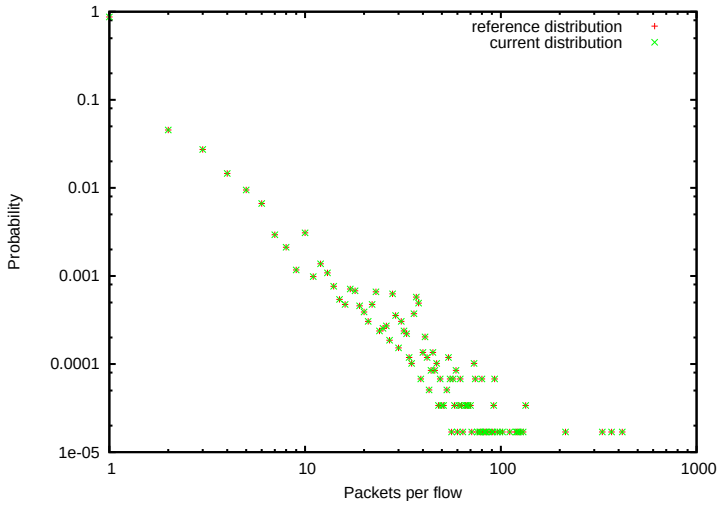


Figure 5.4: Two identical probability distributions — with Relative entropy zero

In figure 5.5 two distributions are shown, the Relative entropy between these distributions can be calculated as being ≈0.0102. It can be seen that although there is a difference between the two distributions. This difference is marginal and doesn't show any indications of an increased packet per flow rate.

5.2.2 Average number of connecting hosts

In section 4.4.2 (page 35) it was said that during a Botnet attack the number of different hosts that connect to the DNS server will be higher than the average amount of connections the server usually has. During normal traffic it can be expected that the number of hosts connecting to a DNS server will have a nice average. By making time slices a baseline could be found that indicates the average number of hosts that connect to a DNS server. This method could be used to detect the detect following attacks:

- DoS attack
- Botnet DDoS attack

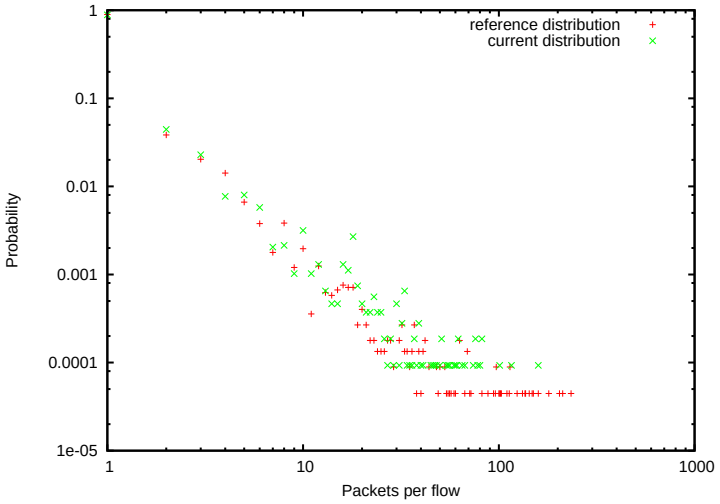


Figure 5.5: Two different probability distributions

For detecting an significant increase in connecting host a threshold needs to be set for every DNS server, to do this equation 5.5 could be used.

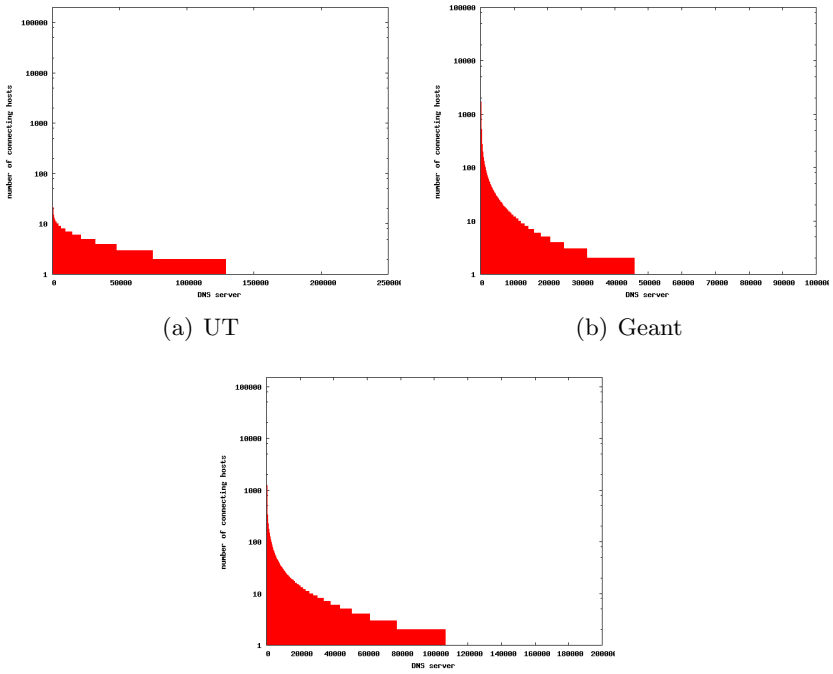
$$C = \begin{cases} 1 & \text{if } x \geq \mu + K * \sigma; \\ 0 & \text{otherwise.} \end{cases} \tag{5.5}$$

The value for K in equation 5.5 needs to be found. When setting the value K to a low value it is possible that normal traffic might be seen as malicious traffic, if the value K to high the smaller attacks will be missed. In Figures 5.6(a), 5.6(b) and 5.6(c) it can be seen that most of the DNS servers are only contacted by less than ten different hosts. It can be expected that the impact of a few more hosts on these hardly contacted DNS servers will give an almost immediate reaction when using equation 5.5, for hosts that are contacted by more hosts this might be less the case.

5.2.3 Average number of flows

The average number of flows a host sends to a DNS server during time slices might increase when an attack occurs. The assumption is that many hosts will have periods of inactivity that will be followed by normal traffic that will only generate a relatively low amount of flows. Only when a host attacks a DNS server will the amount of flows to the DNS server increase to above a certain threshold and can it be seen as possible malicious traffic. This method could be used to detect the detect following attacks:

- DoS attack
- Botnet DDoS attack
- Reflection DDoS attack
- Recursive query attack
- DNS cache poisoning
- DNS tunneling



(a) UT (b) Geant

(c) Surfnet

Figure 5.6: Distribution of connecting hosts

By using equation 5.6 it might be possible to detect the increase in flows during an attack. The average number of flows μ and the standard deviation σ of the number of flows is expected to be reasonable stable over time. The value for K needs to be determined.

$$F = \begin{cases} 1 & \text{if } x \geq \mu + K * \sigma; \\ 0 & \text{otherwise.} \end{cases} \tag{5.6}$$

6 Validation

In this chapter, the validation of the Relative entropy method as described in chapter 5 will be given. Discussion of other methods of search is left as future work.

6.1 Relative entropy validation

In this section, the validation is discussed of the Relative entropy method as described in section 5.2.1. The Relative entropy method was chosen for this validation as it showed an interesting way to correlate an traffic pattern agains "normal" traffic. This type of correlation might make it possible to detect small changes which can indicate an attack.

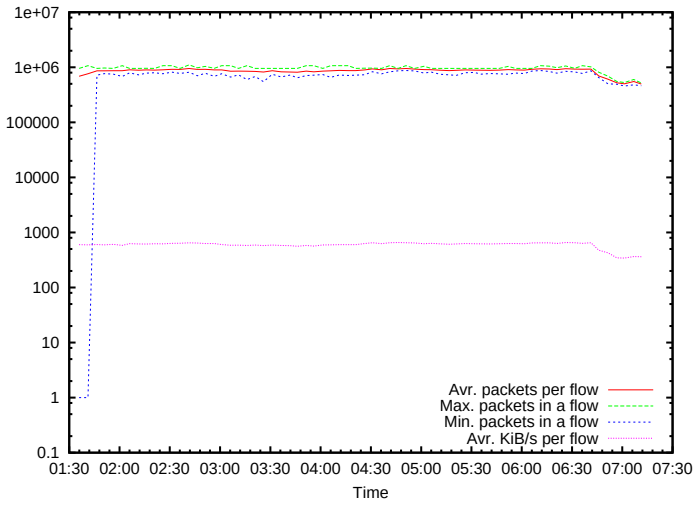
In an initial manual search of the Netflow data, a list of hosts was created ordered by the average amount packets for a flow. For the first two of that list, five minutes time plots were created.

- In figure 6.1(a), shows the time plot of the first host. This host was only contacted during a period of approximately 5,5 hours. In this time, it had an almost stable amount of packets per flow, with about 800000 packets per flow which is very high. Knowing that this server did not respond to the traffic and that it had an average throughput of about 600 KiB/s per flow leads to conclusion this was an attack on this server.

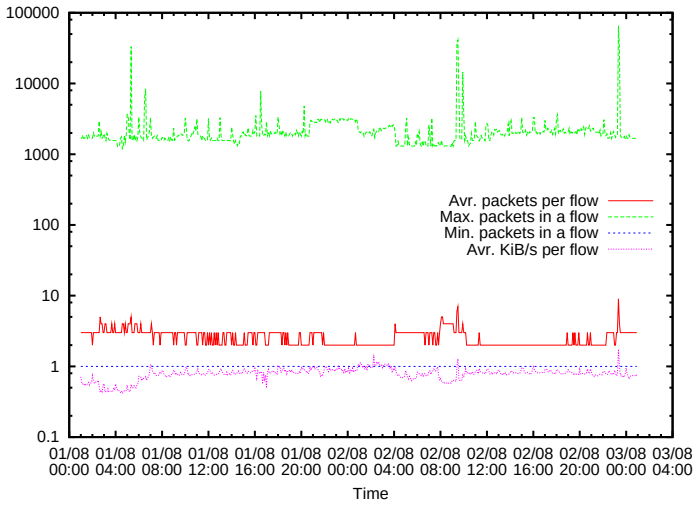
- In figure 6.1(b), shows the time plot for the second host. This host is a normal DNS server. The figure shows that the average amount of packets per flow is low compared to the maximum number of packets in a flow. In the maximum number of packets in a flow line, peaks can be seen, which could indicate an attack. An analysis of these peaks showed that the host causing these peaks did not produce a significant amount of data. The maximum throughput for a flow was calculated at only 50 KiB/s. The host only generated this amount of traffic for approximately one minute.

6.1.1 Relative entropy calculator

For the validation of the Relative entropy method (REM), a program was created. This program calculates the entropy values for traffic destined for port 53 on a host. These calculations are done using equations 5.3, 6.1 and 6.2. These equations originate from equation 5.4. In these equations Ec is the cross entropy, Es is the self entropy, $F(i)$ denotes the probability distribution function for the current time period and $G(i)$ the probability distribution function for the reference period.



(a) Host 1



(b) Host 2

Figure 6.1: Time plots of two hosts with flows having the highest average packets per flow

$$Ec = -\sum_{i \in \mathbb{N}^+} F(i) \log_{10} G(i) \tag{6.1}$$

$$Es = -\sum_{i \in \mathbb{N}^+} F(i) \log_{10} F(i) \tag{6.2}$$

The probability for distribution f and g for bin i can calculated using equation 6.3.

$$F(i) or G(i) = \frac{x_{d,i}}{\sum_{j \in \mathbb{N}^+} x_{d,j}} \tag{6.3}$$

In equation 6.3, $x_{d,i}$ and $x_{d,j}$ are the numbers of flows for bin i where d can be distribution f or g.

When looking at equation 6.1 it can be seen that if $G(i)$ is zero then Ec will be infinite, which is undesirable. To overcome this problem, an assumption is made, namely: there is always a chance for $G(i)$ for a certain i to happen, which means that for a value of i where $G(i)$ should

be zero a value of slightly higher than zero is used. In that case, the total probability of $G(i)$ is marginally larger than one. By using a value of $G(i)$ slightly higher than zero instead, Ec does not go to infinity, but the very low value for $G(i)$ does still have impact in Ec. Equation 6.4 gives the new function for $G(i)$. The initial value $gc = 0.001$ was arbitrarily chosen for the first tests as it denotes a occurrence that is not two frequent but also not to infrequent.

$$G(i) = \begin{cases} \frac{x_i}{\sum_{i \in \mathbb{N}^+} x_i} & \text{if } x_i > 0; \\ \frac{gc}{1+(\sum_{i \in \mathbb{N}^+} x_i)} & \text{otherwise} \end{cases} \tag{6.4}$$

Another problem has to do with the stability of the distributions that are used in the calculation of the Relative entropy. The test outputted hosts with only a few flows which used only one bin which gives a self entropy of zero. The cross entropy of these hosts fluctuated a lot because of the which bin was used in the current distribution. Initial calculations identified about 10.000 DNS servers within the UT domain. With the limitation that a host had to receive at least one hundred flows, the number of DNS servers was reduced to 53.

The Relative entropy program calculates the Relative entropy value in blocks of one hour. Other schema's could be used like: calculating over 5 minute periods or using a sliding window approach. One of the problems with using very small periods is the chance that not enough flows are captured for a meaningful Relative entropy calculation.

The REM requires two probability distribution: one representing the reference and another representing the current distribution. A good set of reference distributions should take the normal fluctuations in traffic into account. The fluctuations can be day and night differences, or differences between week days and weekends. The following list shows possible types of reference distributions that could be maintained:

- **Hourly**: in this case a historic distribution is maintained for every hour of a day.
- **Daily**: for every day of the week a historic distribution is maintained.
- **Week days & weekends**: in this a separation is made between week days and weekends. Two distributions can be saved, for the week day and one for the weekend.
- **Part of the day**: For every part of the day a separate historic distribution is maintained.
- **Sliding window**: By storing the reference distribution of x minutes periods it is possible to use the reference distributions in a sliding window calculation. The value for x should be the same as the Netflow exportation period.

Note that the stored reference distribution is not a probability distribution but contains information as given in table 6.1. The table shows the number of packets in a flow and the corresponding number of flows of that size.

As the reference distribution is made from Netflow data, the possibility exists that the reference distribution contains attacks. The reason for creating a reference distribution in this way is because of the content of DNS traffic. The mixture of hosts that use a DNS server will form the packet per flow distribution. Although this distribution is exponential in nature, it will in most cases not follow a normal exponential distribution curve. So using a normal exponential distribution curve will give high Relative entropy values for normal traffic. An option is to extract those flows that are seen as attacks but this takes time and is hard to achieve, because it is hard to automate. So the simplest and most natural solution is to use the reference distribution from the Netflow data as is.

For the validation of the Relative entropy that will be discussed, the Hourly based historic distribution was used. The use of this reference distribution is because of the precision of looking

Packets per flow	Nr. of flows
1	26681
2	648
3	234
4	241
5	84
6	76
7	36
8	20
9	17
10	9
11	6

Table 6.1: Distribution of packets per flow

at every hourly and less data is used for creating the reference distribution. For example, Daily needs a full week of data to create reference distributions. It is possible that these hours of reference do not contain a lot of traffic, so it is advised to use more days of data to create a complete reference distribution. With the limited data of one week, it is not interesting to use too much for creating the reference distribution, therefore, one day of data was used.

For the updating of the reference distribution, two modes of operation can be used.

1. **Cumulative**: When the reference distribution is updated, the old and new distribution are added up.

2. **Overwrite**: When a new reference distribution can be made, it will override the previous reference distribution.

When during a reference period the Relative entropy value remains below an certain threshold the reference distribution is updated. The value of this threshold is unclear and needs to be calculated.

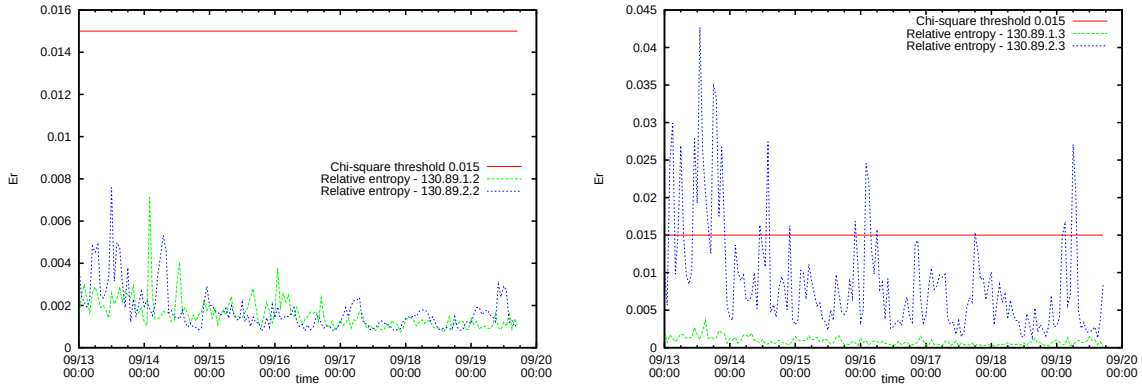
For the validation as will be discussed, the updating was performed cumulatively. By using the cumulative mode, a more complete history is present which should give more realist probability values for $G(i)$. The threshold for accepting a distribution for a new reference distribution was set at 0.3, which is an arbitrarily chosen value.

The validation was done on the Netflow data that was captured in September 2008. This Netflow data was given to the Relative entropy program (REP). The idea was to validate the outcome of REP against the TCPdump data of the UT primary and secondary DNS server. However the volume of data in the TCPdump files and the remaining time for this thesis forced this part of the validation into future work. Because of this the validation was done by inspecting other flow information. The validation was focused only on the UT DNS server but REP did output information about other DNS servers on the UT domain.

6.1.2 Relative entropy output

In this section, the output of REP is discussed. From this discussion, modifications to the Relative entropy calculation will be introduced.

In figure 6.2, the values of Relative entropy can be seen for the two DNS servers. These DNS servers each have two different IP addresses. The figure shows that the Relative entropy of the DNS servers never crosses the threshold of 0.3 and are relatively stable. Karasaridis *et al.* state



(a) Primary DNS server

(b) Secondary DNS server

Figure 6.2: Relative entropy output UT DNS servers ($gc = 0.001$)

that under null-hypothesis the Kullback-leibner distance converges to a chi-square distribution [29]. Using equation 6.5 the value can be calculated for the minimum of the threshold, with α is the false positive rate, c the number of used bins and N the number of samples. This value, however, does not indicate an attack; it just indicates that with certainty 1-α that there is difference between distribution $F(i)$ and $G(i)$. This means that a Relative entropy value lower that should indicate that there is hardly any difference between distribution $F(i)$ and $G(i)$

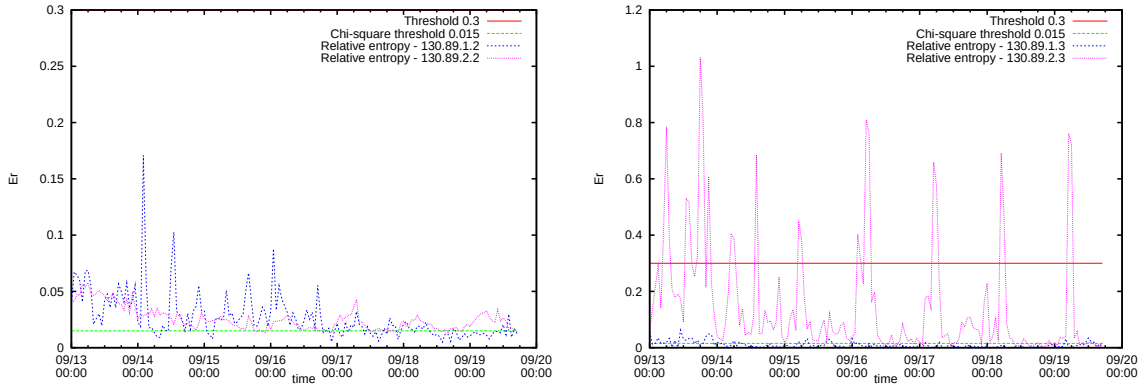
$$THR_{min} = \frac{\chi^2_{1-\alpha,c}}{2N} \tag{6.5}$$

In this situation the minimal threshold can be calculated, using the following parameter values. Because the range of packet per flow is unknown c should be set to infinity. However, simulation shows that only a low amount of bins is actually used, for this reason, $c = 500$ was used. A high certainty of difference is needed, so $\alpha = 10^{-3}$, therefore $\chi^2_{1-\alpha,c} = 603$. Around 40000 samples were captured every hour, at the UT DNS server which means that for these servers $THR_{min} = 0.015$ is an appropriate value. With an increasing sample size, the value for THR_{min} decreases, indicating that very small changes can be detected.

Figure 6.2 shows that the Relative entropy for the UT DNS servers only one crosses this threshold. However, looking at the data, it has known that the other UT DNS servers also show significant difference between distributions. However, it should be expected that they would cross THR_{min} on some occasions. The reason that this does not happen can be found in equation 6.4 with $gc = 0.001$. The factor $gc = 0.001$ is too large as in equation 6.1 this would lead to a very small impact value for i for low probability values of $F(i)$. By setting the value of gc to 1^{-100}, the impact of low probability values of $F(i)$ increases. In figure 6.3 the outcome of the Relative entropy function can be seen for the $gc = 1^{-100}$. Higher fluctuations in the Relative entropy can be seen, which is consistent more consistent with the data.

For one the DNS server(139.89.2.3) also the initially set thresholds of 0.3 is crossed several times. Investigating these events shows that the packet per flow distribution has changed during these periods. Figure 6.4(a) shows one of the distributions comparisons. Although there is difference between the distributions, this most likely is not an attack. Two possible explanations can be found, namely: the threshold of 0.3 is too low or the presence new packet per flow values in the current distribution have a large impact on the Relative entropy.

As stated earlier, the threshold of 0.3 is a arbitrarily chosen value. Knowing that, for the distributions in figure 6.4(a), the Relative entropy is 0.39. Therefore it is likely that the threshold of 0.3 might be too low. Figures 6.4(b), 6.4(c) and 6.4(d) show a comparison between



(a) Primary DNS server

(b) Secondary DNS server

Figure 6.3: Relative entropy output UT DNS servers ($gc = 1^{-100}$)

the distribution at a time period and the reference distribution with which it was compared during the Relative entropy calculation. These figures show that with an increasing Relative entropy value, the difference between the current distribution and the reference distribution becomes more clear.

The figures 6.4(a), 6.4(b), 6.4(c) and 6.4(d) show that frequently the current distribution occupies bins that were not used in the reference distribution. The occurrences have an impact on the Relative entropy. It could be expected that the impact of these new packet per flow values would be similar for every distribution. However in figure 6.5 increased traffic is shown with new high packet per flows bins being present. This period should have a Relative entropy value higher than 0.3, however it only has a value of 0.228. The reason for this lies in the probabilities. The probabilities for these new values are low which results in the impact of these value is also lower. The reason that these probability values are low is because of the high probability for one packet per flow.

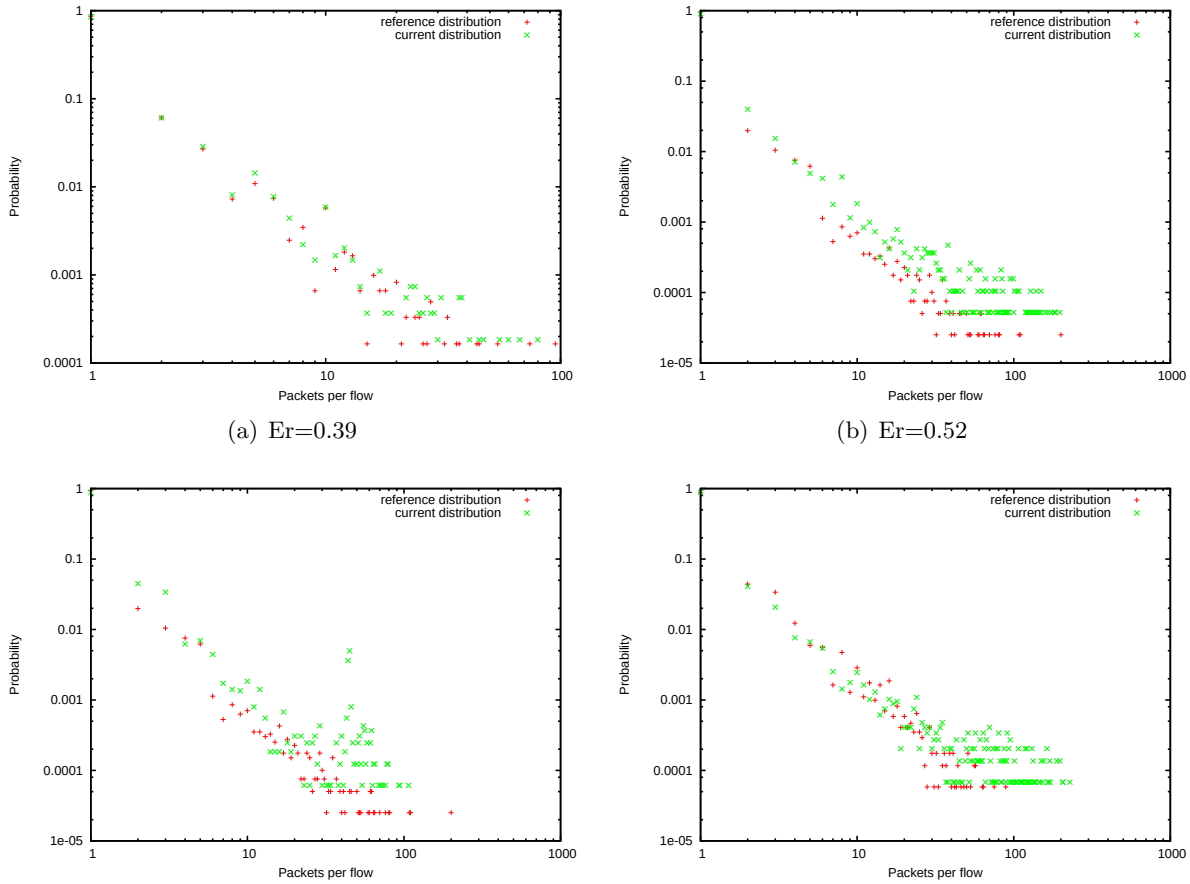
One other extreme can be calculated using an example. In table 6.2 two distributions are shown. These distributions are by definition not equal. The "current" distribution is a shifted version of the "reference" distribution. Therefore there is one "packet" less is in the "current" distribution. So these distributions are not equal and would give a large Relative entropy value, but surely does not show an attack.

Reference	1	0	1	0	1	0	1	0	1
Current	0	1	0	1	0	1	0	1	0

Table 6.2: Example distributions — High entropy but no attack

$$Es(i) = F(i) \log_{10} F(i) \tag{6.6}$$

Figure 6.6 shows the output of equation 6.6 which indicates the impact for a probability $F(i)$ of the self entropy formula. It can be seen in the figure that with a decreasing probability, the impact lowers. It also shows that when the probability increases the impact will also decrease. It can be expected the with a exponential distribution that there will be a set of packet per flow bins with a low to very low probability (1^{-5} and smaller). Low probability bins have a significant impact only when enough of these bins exist. The amount of bins, however, depends heavily on the value of the probability, the lower the probability, the higher the number of needed bins.



(a) Er=0.39

(b) Er=0.52

(c) Er=0.76

(d) Er=1.03

Figure 6.4: Distribution comparisons

The impact of high packet per flow bins with a low probability might not be enough, for this reason, modifications for calculating the Relative entropy were made. These modifications are:

- **Separating probability domain**: In this modification the distribution domain is separated into equal ranges of packet per flow bins. In figure 6.7, this separation is illustrated with a range five bins per probability domain. By separating the distribution, the impact of the probabilities change. Where a range would normally have a very low probability, in this situation, it would have a much higher probability. Mathematically, this is represented in equation 6.7.

$$Er_{separate} = -\left(\sum_{i=1}^{5} iF(i)_{[1,5]} \log_{10} \frac{F(i)_{[1,5]}}{G(i)_{[1,5]}} + \sum_{i=6}^{10} iF(i)_{[6,10]} \log_{10} \frac{F(i)_{[6,10]}}{G(i)_{[6,10]}} + ...\right) \quad (6.7)$$

- **Bin merging**: In this modification, the distributions domain bins are mapped to another distribution domain. This mapping is done on the basis of the equation 6.8. As can be seen in the equation, for an increasing i, more bins are mapped to the same i_{merge}. Mathematically, the probability distribution function is changed, so equation 6.3 becomes equation 6.9. The resulting Relative entropy calculation is represented in equation 6.10. In figure 6.7 the effect of bin merging is illustrated.

$$i_{merge} = \lfloor \sqrt{i} \rfloor \quad (6.8)$$

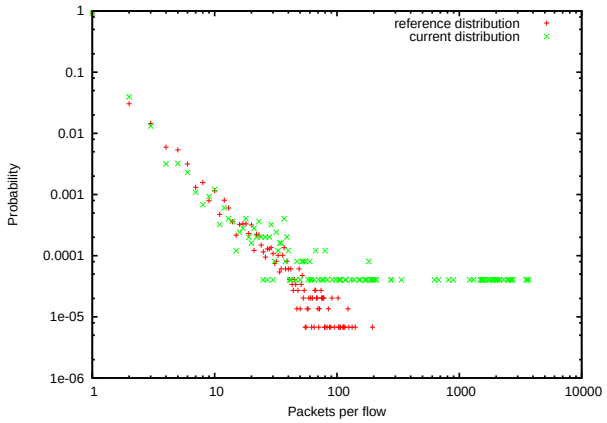


Figure 6.5: Distribution comparison — Er=0.228

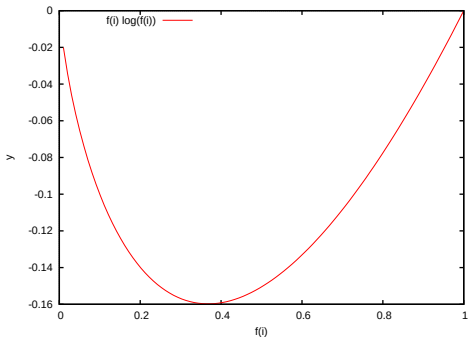


Figure 6.6: Probability impact

$$G_{merge}(i) or F_{merge}(i) = \frac{\sum_{1+(i-1)^2}^{i^2} x_{d,i}}{\sum_{j \in \mathbb{N}^+} x_{d,j}} \tag{6.9}$$

$$Er_{merge} = -\sum_{i \in \mathbb{N}^+} F_{merge}(i) \log_{10} \frac{F_{merge}(i)}{G_{merge}(i)} \tag{6.10}$$

The use of *bin merging* should decrease the use of *gc* during the calculation of the Relative entropy. Also the impact of new bins, as was shown earlier, should be reduced, as neighboring bins could contain flows.

- **Index dependency**: In this modification, the formula for the Relative entropy depends on i. In equation 6.11, shows equation 5.3 with the added dependency of i. Other formulas for making the Relative entropy dependant on i may exist that might perform better. However, these are out of the scope of this research.

$$Er_{depend} = -\sum_{i=1}^{N} iF(i) \log_{10} \frac{F(i)}{G(i)} \tag{6.11}$$

- **Remove most probable bin**: By removing the largest packet per flow bin from the entire Relative entropy calculation, the impact of bins with a low flow counts is increased. The removal of this bin should be done on the basis of the reference distribution as the largest packet per flow bin should be the same bin in both the reference distribution and the current distribution. If it is not the same bin, then a clear difference in Relative entropy should be visible.

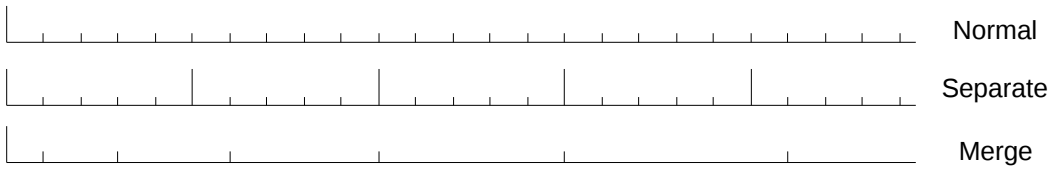


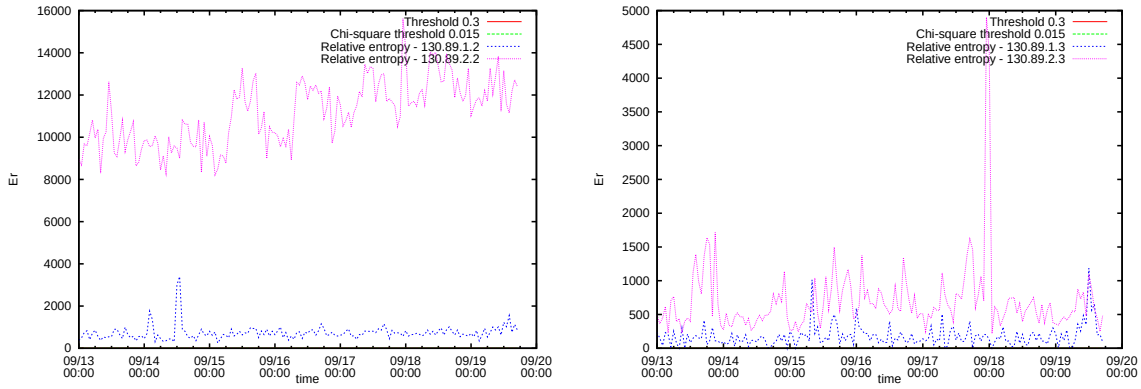
Figure 6.7: Relative entropy modifications

For each of the modifications, calculations where performed. As the impact on the Relative entropy value is unknown for these modifications the threshold of 0.3 was maintained during these calculations.

Separating probability domain

In figure 6.8, the Relative entropy time plots of the DNS servers for the probability domain separation is shown. The probability domain was split into parts of 10 bins. The values for the Relative entropy are much larger than with the original function; an increase in the Relative entropy for certain situations was the goal. However it is strange to see that 130.89.2.2 has much higher overall Relative entropy. It indicates there is a lot of difference between the distributions, however this is not visible in figures 6.2 and 6.3 nor is it in the data.

One thing that can be said about separating the probability domain is that there is a loss of connectivity between the part of the separated domain. As an example, table 6.3 shows a distribution. When using probability domain separation, two domains with 5 bins can be created. For the second domain, the Relative entropy impact is 0.20. When using the original calculations this part only has an impact of 0.03, which is almost a factor 7 smaller. When first domain would have more samples, the impact would remain the same for the domain separation, but in original calculation the impact would decrease rapidly, which is what needs to happen. So this loss of connectivity might has unwanted effect that very small changes have a disproportional large impact on the Relative entropy. A reason that for the line 130.89.2.2 in figure 6.8(a) is that for higher values of i, there are more differences in used bins in the current distribution and the reference distribution. Splitting the domain differently can be done to increase the effectiveness of the domain separation. For example a larger more bins per domain could be effective, or using a larger domains for the low packers per flow value range.



(a) Primary DNS server

(b) Secondary DNS server

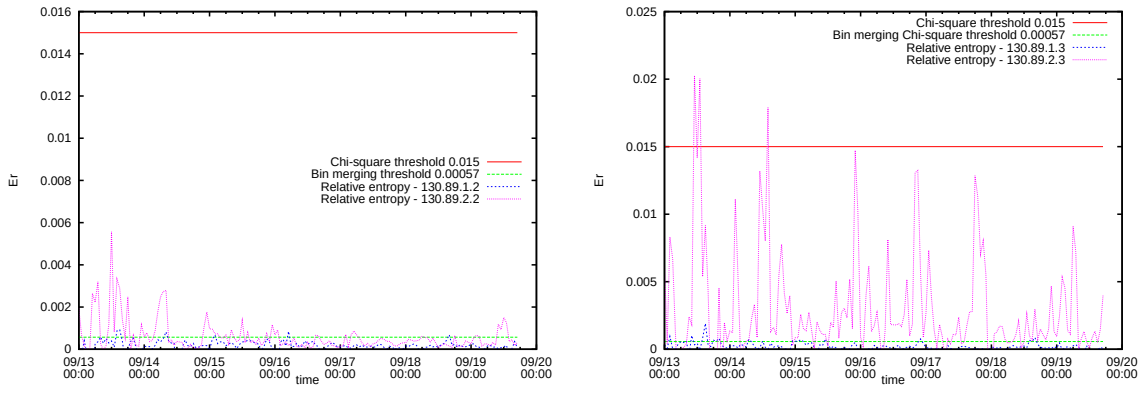
Figure 6.8: Relative entropy — probability domain separation

Samples										
Reference	9	4	2	1	1	1	1	1	0	0
Current	9	4	2	1	1	1	2	0	0	0
Probabilities										
Normal (Reference)	$\frac{9}{20}$	$\frac{4}{20}$	$\frac{2}{20}$	$\frac{1}{20}$	$\frac{1}{20}$	$\frac{1}{20}$	$\frac{1}{20}$	$\frac{1}{20}$	0	0
Normal (Current)	$\frac{9}{20}$	$\frac{4}{20}$	$\frac{2}{20}$	$\frac{1}{20}$	$\frac{1}{20}$	$\frac{1}{20}$	$\frac{2}{20}$	0	0	0
Separate (Reference)	$\frac{9}{17}$	$\frac{4}{17}$	$\frac{2}{17}$	$\frac{1}{17}$	$\frac{1}{17}$	$\frac{1}{3}$	$\frac{1}{3}$	$\frac{1}{3}$	0	0
Separate (Current)	$\frac{9}{17}$	$\frac{4}{17}$	$\frac{2}{17}$	$\frac{1}{17}$	$\frac{1}{17}$	$\frac{1}{3}$	$\frac{2}{3}$	0	0	0

Table 6.3: Example distribution — Domain separation

Bin merging

In figure 6.9, the Relative entropy time plots of the DNS servers for the bin merging modification can be seen. Peaks that where present for certain hours in figure 6.3 have now disappeared. It can be seen that the threshold of 0.3 is never crossed and the calculated threshold of 0.015 is crossed only a small number of times. One could state that the outcome of this calculation shows that this method is unusable, however the threshold values need to be adjusted for the new situation. In the implementation, the number of bins was limited to 500, but only about 10 to 20 where every used. Making a recalculation using equation 6.5 with $\alpha = 10^{-3}$, $c = 20$ and $N = 40000$ shows that a Relative entropy value of 0.00057 should already indicate a change between the distributions. In figure 6.9, this threshold is drawn and it can be seen that a large amount periods cross this threshold. Although all of the peaks show a clear difference between the reference and current distributions most of these difference showed a decrease and not an increase in packets per flow. Figure 6.10, for example, shows the distribution plot where there is a clear decrease in packets per flow, but it has a Relative entropy of 0.015, which is the fourth highest peak that can be seen in figure 6.9 which is clearly unwanted.



(a) Primary DNS server

(b) Secondary DNS server

Figure 6.9: Relative entropy — Bin merging

Index dependency

In figure 6.11 the Relative entropy time plots of the DNS servers for the index dependency modification as calculated can be seen. The figure shows two large peaks for one IP address. The largest of these peaks was found for the distribution in figure 6.5, which was not found using

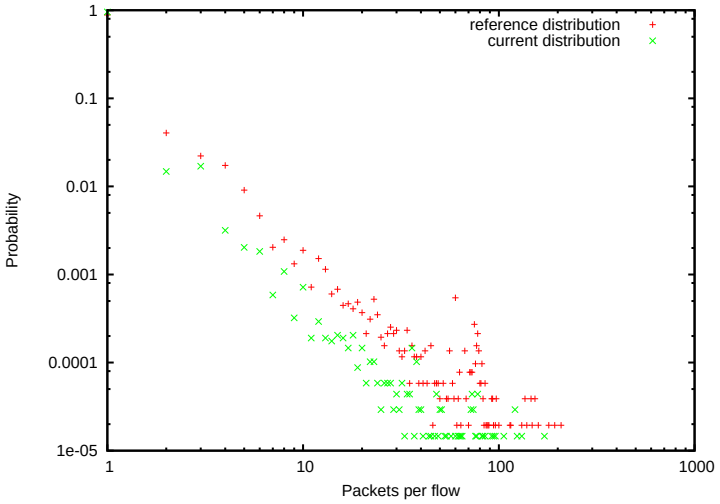
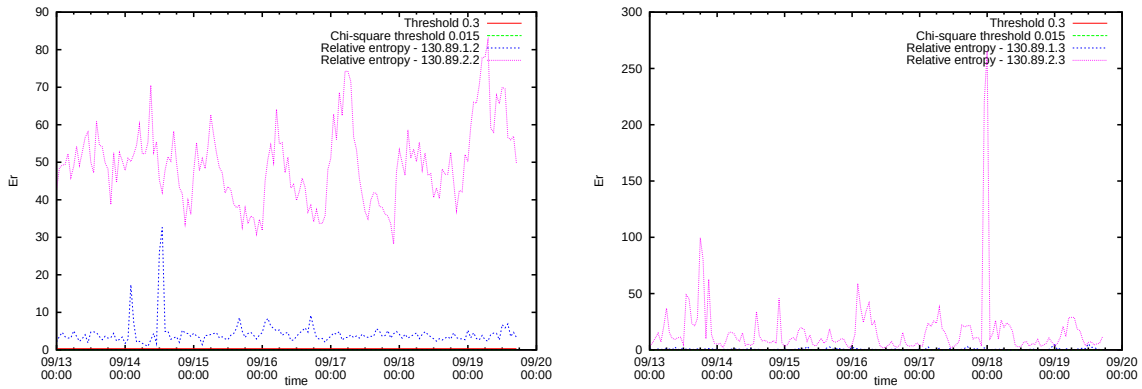


Figure 6.10: Distribution comparison — Re=0.015

the normal REM. The other peak also showed a an increase but was already captured by the original calculation. These peaks show an clear difference between the distributions. There is, however, one line 130.89.2.2 which has a high Relative entropy throughout the time series. For the *probability domain separation* modification, this larger Relative entropy for line 130.89.2.2 was said to be possibly caused by difference in the use of higher value i bins compared to the reference distribution. This reasoning also holds in this calculation. Because of the added factor i in equation 6.11, these previously unused bins will have a larger impact. This might indicate normal traffic, but it could also indicate the presence of an attack.



(a) Primary DNS server

(b) Secondary DNS server

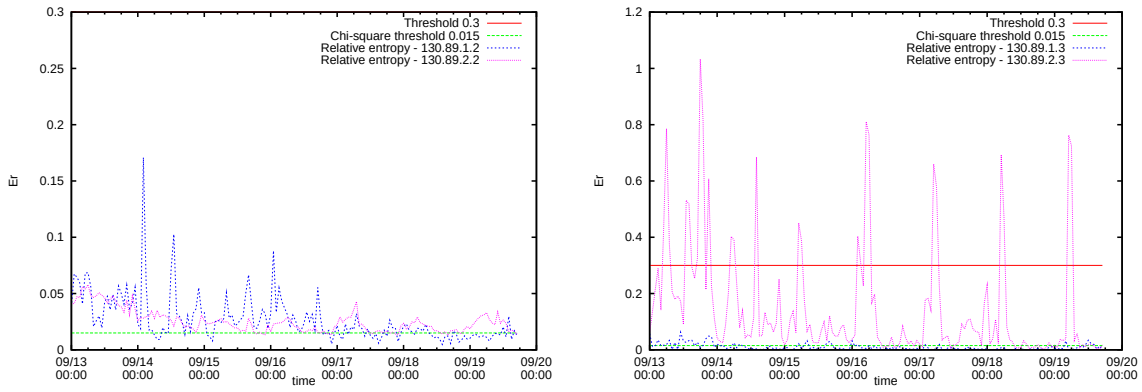
Figure 6.11: Relative entropy — i impact

Remove most probable bin

In figures 6.12(a) and 6.12(b) the Relative entropy time plots of the DNS servers for the *remove most probable bin* modification is shown. When comparing figure 6.12 with figure 6.3 no real differences can be seen in the outcome. The values of the Relative entropy shows that there was a difference, but it was only marginally.

There is one negative side effect to removing this most probable bin: using the original REM, it is possible to detect large changes to even the most probable bin. This could happen if an

attacker continuously spoofs its source IP address.



(a) Primary DNS server

(b) Secondary DNS server

Figure 6.12: Relative entropy — Remove top packet per flow bin

6.1.3 Threshold calculation

In previous section, some modifications to the Relative entropy calculation were introduces and the output of these modifications given. The validation of each of these modification and the original Relative entropy calculation all depend on the possibility to calculate a good threshold. Karasaridis *et al.* state two functions to define a value range for the threshold [29]. Equations 6.12 and 6.13 show these formulas.

$$E(n|f) \int f \ln(\frac{f}{g}) \geq \beta \ln(\frac{\beta}{1-\alpha}) + (1-\beta) \ln(\frac{1-\beta}{\alpha}) \tag{6.12}$$

$$E(n|g) \int g \ln(\frac{g}{f}) \geq \alpha \ln(\frac{\alpha}{1-\beta}) + (1-\alpha) \ln(\frac{1-\alpha}{\beta}) \tag{6.13}$$

These equations are based on Kullbacks definition of the type 1 and type 2 errors [34]. The right hand side denotes the type 1 error respectively the type 2 error. For these error types the α represents the false positive probability and β the false negative probability. $\int f \ln(\frac{f}{g})$ denotes the entropy calculation. In these equations, $E(n|f)$ and $E(n|g)$ denote the expected number of flows. The expected number of flows is the number of flows needed to show a difference. So to see large deviations, only a small number of flows should already suffice. If more subtle differences are part of the focus, a much higher number of samples is needed. Equations 6.12 and 6.13 show that if a higher value for $E(n|f)$ respectively $E(n|g)$ is used, the threshold for the Relative entropy will decrease. In table 6.4 threshold ranges are given for different values of $E(n|d)$, where d is either distribution f or g. For these calculations the following parameters where used: $\alpha = 10^{-2}$ and $\beta = 10^{-3}$. These parameters are the same as in the work by Karasaridis *et al.*. With an $E(n|d)$ of 1000, the threshold is lower than the chi-square threshold.

$E(n\|d)$	10	100	1000
$\int f \ln(\frac{f}{g}) \geq$	0.46	0.046	0.0046
$\int g \ln(\frac{g}{f}) \geq$	0.68	0.068	0.0068

Table 6.4: Lower and upper bounds for Relative entropy threshold

Although Karasaridis *et al.* state that equations 6.12 and 6.13 are correct, it is not clear if that is the case. Although Kullback states almost similar equation in the book "Information Theory and statistics", it has not become clear if these are applicable here [34]. The equations are therefore taken to be correct. As the high of the Relative entropy values for the modifications differs greatly from the original Relative entropy calculation, the threshold equations might need to be changed.

- **Separating Probability domain**: Equation 6.7 is clearly a different calculation for entropy, so from that alone, it could be expected that the thresholds from equations 6.12 and 6.13 would not be correct. A look at the Relative entropy values in figure 6.8 shows that these are for all four servers a lot higher than the normal Relative entropy calculation. This clearly shows that a different threshold calculation is needed for this modification.

- **Bin merging**: For *bin merging*, the original Relative entropy is used and only the distributions are changed. This change in distribution should be the only thing that has any influence. However, the equations 6.12 and 6.13 are not dependant on the distribution. In section 6.1.2, a minimum Relative entropy threshold was calculated which is much lower than the originally calculated value. From this it should be expected that the equations 6.12 and 6.13 need to give lower threshold values. However equations 6.12 and 6.13 are not dependant on the number of bins so it will not do so.

- **Index dependency**: When making the Relative entropy formula dependant on i it is clear that this dependency also needs to represented in calculation of the threshold. From equation 6.11 it can be expected that the threshold will be higher due to the addition of dependency on i.

- **Remove most probable bin**: As with *bin merging* the calculation of the Relative entropy is not changed. Only the distributions over which the Relative entropy is calculated are changed.

So for the modifications *Separating Probability domain* and *Index dependency* a different calculation is needed in order to calculate a correct threshold range. For the other modifications, the original should be usable. It is, however, unclear what actual changes need to be made in order to correctly calculate the threshold value range for all the modifications. This is left as future work.

For instance, for the *index dependency* modification, it might be needed to make equations 6.12 and 6.13 dependant on $E(i)$. For an exponential distribution, the $E(i)$ is a value which can be calculated using $\frac{1}{\lambda}$. For an increasing λ the value for $E(i)$ will decrease. With the addition of $E(i)$ the equations 6.12 and 6.13 for the *index dependency* modification might look like equations 6.14 and 6.15. The value that needs to be chosen for $E(i)$ will most likely be smaller than $\frac{1}{100}$ but the actual value is unknown.

$$E(n|f)E(i)\int f\ln(\frac{f}{g}) \geq \beta\ln(\frac{\beta}{1-\alpha}) + (1-\beta)\ln(\frac{1-\beta}{\alpha}) \tag{6.14}$$

$$E(n|g)E(i)\int g\ln(\frac{g}{f}) \geq \alpha\ln(\frac{\alpha}{1-\beta}) + (1-\alpha)\ln(\frac{1-\alpha}{\beta}) \tag{6.15}$$

For the *bin merging* modification, more research is needed to define a usable threshold value.

6.2 Summary

In this chapter an attempt was made to validate the usage of the REM on packets per flow information. From the tests that where done it has become clear that the normal Relative entropy formula in combination with packet per flow information has some issues which need to be addressed. Some possible solutions for dealing with these issues have been formulated and tested. The *index dependant* modification correctly shows an increased packet per flow usage for certain periods. While the *bin merge* modification show a decreases in traffic which is unwanted. So not all of the modifications had the intended outcome. By combining the *index dependant* and *bin merge* modifications a better results might be possible but this is left as future work. The thresholds that need to be calculated give a high certainty of the existence of an attack. The modifications to the Relative entropy calculations need a different threshold calculation.

7 Conclusions and Recommendations

In this chapter, a summary will be given of the contents of this thesis, and conclusions will be drawn. The final part lists future work that was identified from this thesis.

7.1 Conclusions

The state of the art on NIDS systems shows that work has been done using Netflow to detect attacks on DNS servers. How well these detection methods actually work is unknown. A literature study was done to find information about the DNS service, the traffic DNS servers receive and the sorts of attacks that are carried out. A statistical analysis was done to find several types of DNS traffic anomalies in TCPdump data from the UT DNS servers.

An overview was created of all the possible attack involving DNS servers. The characteristics of these attacks have been identified and described. A translation was made from these characteristics to algorithms for Netflow. For some of these characteristics, matches could be found in different attacks, which has lead to more general search methods. One of these search methods, which uses Relative entropy, was implemented and tailored so that results about its usability could be found.

The statistics, have shown that only a about 2-3% of the traffic on the Internet is caused by DNS. The analysis of the TCPdump data showed that 80% of the DNS queries were repeated questions. It was also shown that there are clients that have a bad PRNG, which results in bad choices for query IDs and source port numbers. This makes them vulnerable to attacks. From these actions by the DNS clients it can be concluded that some DNS client software contains issues that need to be resolved.

For the DNS servers it can be seen that the PRNG is not ideal, because the query ID distribution is not uniform. For the distribution of the source port number a clear uniform distribution was shown. The existence of *repeated*, *identical* and *A of IP* queries shows that DNS servers, although better than DNS clients, still have some security and protocol implementation issues.

The amount of traffic that is captured for a specific source or destination makes detecting certain attacks hard. For a DoS attack, the amount of traffic can be high, but a DoS attack could also make the DNS software crash by exploiting software bugs. This would result in only a few packets performing a successful DoS attack. A low amount of traffic can only give information about the presence of a port scan. For all other attacks, it is much harder to give conclusive results when the amount of captured traffic is very low.

The research questions stated in section 1.1 on page 1.1 can now be answered.

What are the types of attacks that target DNS servers, and in what way?

With the prominent role the DNS system plays in the Internet, a lot of effort has been put into defending it. A literature study showed that seven types of attacks exist, most of which target a DNS server to deny normal service to its clients. But interestingly enough, an attack exists that uses a DNS server to attack a host. Furthermore, it is possible to use the DNS service for covert data exchange.

What are the traffic characteristics of the attacks and how do they translate to Netflow data?

For all of the attacks, the combination of the characteristics is a key to identifying a certain attack in Netflow data. However, for *buffer overflow* attacks, the content of a packet is the best characteristic to identify the attack. More general methods of detection are possible because several attacks have the same characteristics. For instance, DoS attacks and DNS tunneling will be visible due to an increase in traffic.

Which algorithms can be used to successfully detect attacks?

In this thesis, some general methods of detection have been described, but only one has been implemented and analyzed. As none of the other methods has been implemented, their effectiveness is unknown. Some of these methods might not work at all, or they need to be changed in some way.

The Relative entropy method looks like a promising way to detect attacks, as it shows the difference between a stored normal situation and observed time frames in Netflow data. It has been shown that a straightforward implementation of the Kullback-Leibner distance has problems that need to be addressed. The biggest of these problems being the lack of impact of flows with large packet counts on the Relative entropy value. The modified Relative entropy implementations, as described in this thesis, show various ways to overcome these problems. The outcome of tests on these modifications shows that some things can be done to improve on the original Relative entropy calculation. For instance, by using the *index dependant* modification, an increasing index of packets per flow bins have an increasing impact on the Relative entropy value.

What are the false positives probabilities of the algorithms that can be used?

The outcome of this research question is inconclusive. The effectiveness of the Relative entropy method also partially depends on the ability to define a proper threshold. If this threshold is set to a low value, the probability for false positive events will increase. For the normal Relative entropy calculation, it is assumed that the equations given by Karasaridis *et al.* are correct, although their correctness is unclear and warrants future research. From this, the correct formulation of the lower bound threshold for the altered Relative entropy calculations is hampered. No conclusion can be given on the false positive probabilities of the Relative entropy method.

What are the false negatives probabilities of the algorithms that can be used?

The outcome of this research question is inconclusive. This also depends on the ability to define a proper threshold. If the threshold is set to a high value, some attacks may go unnoticed. For the altered Relative entropy calculations, the correctness of the high bound threshold calculations

are unknown. No conclusion can be given on the false negative probabilities of the Relative entropy method.

How can attacks on DNS servers be detected by using Netflow data from routers?

From several manual searches in the Netflow data, it has become clear that it is possible to detect attacks on DNS servers based on Netflow data alone. The research in literature also showed that attacks on DNS servers can be found in Netflow. How well the attacks can be detected, depends on the methods that were used to detect them. Several manual searches in the Netflow data have shown that the number of packets per flow can be used for detecting DNS anomalies. From the research that was done in this thesis on the Relative entropy method, it was found that the method has probable application. However there are some issues that need to researched to make the method usable. The changes to the Relative entropy calculations that were described in this thesis show that there are ways to overcome the issues of Relative entropy for attack detection. However no conclusion can be made on what the threshold should be for excepting an anomaly in these alternated calculations.

7.2 Future work

In this thesis, a lot of work was done, but not all questions could not be answered conclusively. Therefore, the following topics for future work were identified:

- The statistics gathered for the TCPdump files indicate that some clients act erroneous. **Which clients have security and protocol issues and what are they doing wrong?** This is an interesting question because these clients cause an increased load on a DNS server and are more vulnerable to exploits.

- Moore *et al.* found that only a very small percentage (0.5%) of all DoS attacks are targeting DNS servers. As not all attacks on DNS servers are DoS attacks, it could be expected that the actual number of attacks on DNS servers is higher. The question however is: **How high is the number of attacks involving DNS servers?**

- In chapter 5, several methods of search have been described but have been left as future work. **How usable are these methods and how good are they?**

- **What other detection methods are possible and how do they perform?** The detection methods that were described in chapter 5 might not be the only methods that are possible.

- Before it can be definitively said if the Relative entropy method works, some unresolved issues need to be addressed, namely:

  - **How to increase the impact of flows with a large number of packets and a very low probability of occurrence?** Although the basis of this has been discussed in this thesis, there might be other methods of doing this.
  - **How to deal with shifts in packet per flow bin usage?** In this thesis some methods of removing this problem have been proposed, but better alternatives might well be possible.

- **The calculations for determining a proper threshold value for the Relative entropy method need to be proven.** Without this calculation it is hard, if not impossible, to make a working system that can find attacks with a high probability, and not overwhelm analysts with data to analyze because of false positives.
- Although some data was found suggesting that the Relative entropy method can be used for anomaly detection, a true validation has not been preformed. The best validation might be done on the basis of traffic captured at the same point in the network. **How good can the Relative entropy method perform?**
- In the research that was done for hosts that received less than 100 flows, no Relative entropy was calculated. The question, however remains: **With how many flows is a distribution stable enough to perform a Relative entropy calculation?**
- If the reference distribution is attack-free, the Relative entropy methods performance could increase. The validation as done in this thesis might very well have used reference distributions with attacks in them. So the question is: **What would be a better way to create an initial reference distribution?**
- **What influence does sampling have on the effectiveness of the Relative entropy method?** The validation that was done in this thesis only focused on unsampled Netflow data, so the effect of sampling is unknown.

Appendices

A

DNS requests and their responses

In this appendix the possible messages that can be send to the DNS server will be put to the test, and an analysis of their responses is made. It is clear that every request will have an appropriate response, but every DNS server implementing might react differently to the same request.

To identify the responses a separation needs to made between TCP traffic and UDP traffic and within that a separation needs to be made for the different types of requests that can be done. For this analysis the use of a packet manipulator was used, namely the Python packet called Scapy [7]. The traffic generated was captured and analyzed using wireshark [15]. The DNS server used as a target for the malicious packets had BIND installed as DNS service software [55].

A.1 UDP traffic

In UDP queries by the client are done by sending one packet to the DNS server. The only things that can make the packet size differ are the option fields of the IP header and the size of the DNS query. To send a UDP DNS packet using Scapy the following code was used:

```
import sys
from scapy import *
conf.verb=0

i = IP()
u = UDP()
d = DNS()

i.dst = "130.89.2.3"

u.dport = 53
u.sport = 31337

d.id = 31337
d.qr = 0
d.opcode = 0
d.qdcount = 1
d.qd = '\x00\x00\x01\x00\x01'
```

```
packet = i/u/d
sr1(packet)
```

As can be seen all three layers (IP, UDP, DNS) are initialized and configured. In this example the root DNS domain is queried for; the response to this is a list with all root DNS servers. What will a DNS server respond with when there is no question set in the UDP packet. This can be done setting *d.qd* to ". The response to this question is a UDP packet with the format error flag in the DNS header set high. the size of this response message is 40 Bytes.

According to RFC1034 it is allowed to send multiple questions to a DNS server in a single packet [41]. The presence of the *d.qdcount* variable indicates that indeed more than one question can be asked per query, to send this the following settings where used:

```
d.qdcount =2
d.qd='\x03com\x00\x00\x01\x00\x01\x03net\x00\x00\x01\x00\x01'
```

In this request the names *.com* and *.net* are asked to be resolved. As the DNS protocol describes the dot needs to be replaced by a hexadecimal number representing the length of the name till the next dot. The response to this message was not what should be expected when reading RFC1034. The response to this was a DNS packet without an answer and with the format error flag set high. The reason for this format error in because after RFC1034 and RFC1035 where adopted it was realized that there was a flaw, namely the RFCs did not define what should happen when more than one question was asked and one of them contained an error [59][42]. Because of this flaw most DNS server software treat a DNS packet with multiple questions as a format error, in this way forcing the client to rephrase its question.

By the definition of RFC1034 DNS traffic should switch over to the use of TCP if either the request or the response is bigger than 512 Bytes [41]. Another parameter boundary given by RFC1034 is the maximum for the size of a query name (QNAME); the maximum size of a queryable name should be 255 Bytes. By setting the *d.qd* parameter to the appropriate value it was verified that indeed the maximum size of the QNAME variable is 255 Bytes. The response from the DNS server on receiving a QNAME of more than 255 Bytes is a 40 Bytes packet with the server failure flag set high.

From the responses that were received during these tests it is obvious that when the server cannot handle a request it will respond with a DNS response with one of the error flags set high. The size of this packet will be 40 Bytes.

In figure A.3(a) the distribution domain of a DNS UDP packet can be seen.

A.2 TCP traffic

In figure A.1 the sequence of packets can be seen for a DNS resolution using TCP. In this stream of packets there are packets with the same information as would be set in an UDP based query. The only difference between those UDP and TCP are the UDP and TCP header. By removing these already discussed packets from the equation the only packets that will be left are the TCP control messages. The responses that can be gotten from sending certain types of packets will be described in this section.

Figure A.2 depicts the TCP header. It can be seen that TCP has a 8 bit flag field. The flags in this field have the following meaning:

- Congestion Window Reduced (CWR): This field is used in the Explicit Congestion Notification (ECN) protocol to indicate that the client has reduced its TCP congestion window [49].

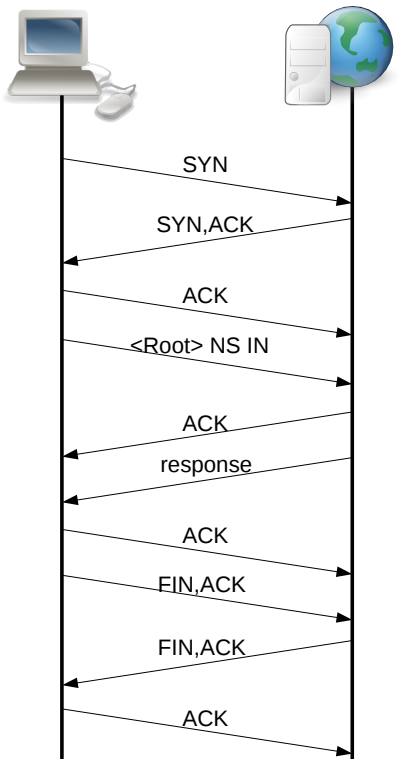


Figure A.1: Sequence diagram of DNS connection over TCP

0 0	1	2	3	4	5	6	7	8	9	1 0	1	2	3	4	5	6	7	8	9	2 0	1	2	3	4	5	6	7	8	9	3 0	1
Source Port																Destination Port															
Sequence Number																															
Acknowledgment Number																															
Data Offset				Reserved						U R G	A C K	P S H	R S T	S Y N	F I N	Window															
Checksum																Urgent Pointer															
Options																													Padding		
Data																															

Figure A.2: The IPv4 header

- ECN-echo (ECN): To indicate that CE Congestion Experienced was received by the client.
- Urgent (URG): This flag indicates that this packet needs to be processed immediately.
- Acknowledgment (ACK): This flag indicates the successful reception of previous packets.
- Push (PSH): This is intended to indicate to the server that this packet needs to handled

Request	Response
SYN	SYN, ACK
SYN, ACK	RST
SYN, RST	no response
FYN, SYN	SYN, ACK
ACK	RST
FIN, ACK	RST
RST	no response
FIN	no response
FIN, SYN, ACK	RST
FIN, SYN, ACK, RST	no response
none	no response

Table A.1: The TCP flags of request and corresponding response using BIND

as soon possible.

- Reset (RST): A packet with this flag set indicates that the connection needs to be reset.
- Synchronization (SYN): This flag is used in the three-way handshake to synchronize the two peers of the connection.
- Finish (FIN): This flag is used in the connection teardown to indicate that a side wants to close the connection.

In Table A.1 the response message to every type of TCP control message can be seen. The responses are to the first message that is send during a TCP session. Other response can be expected for some requests. The other TCP flags that can be set have no visible difference to the same packets without these flags and have therefore been excluded from table A.1. It can be seen that the reaction to most packets is either a TCP RST packet or no response. During other states of the TCP protocol other reactions might be expected.

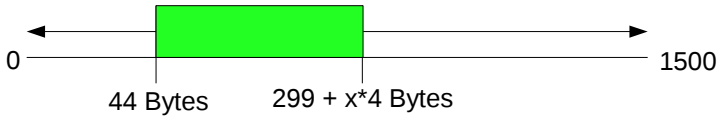
The TCP control messages have a minimum size of 40 Bytes, the size of the packets can increase when there are additional IP and/or TCP option fields set. All of these option fields increase of the size of a packet by 4 Bytes per option field.

By using Wireshark it can be shown that a minimal DNS query using TCP is 59 bytes, however this cannot be taken as a minimum because the average number of octets per packets in a flow will be lower because of the TCP control packets that have only a size of 40 bytes. The minimal average number of octets per packet in a flow can be used as a lower bound threshold. An explanation for this can be given as follows: the absence of a TCP control message in a flow would increase the average. the absence of the DNS query will make the average 40 bytes plus a multitude of 4 bytes. The absolute minimal average for a TCP connection to a DNS server can be calculated as follows:

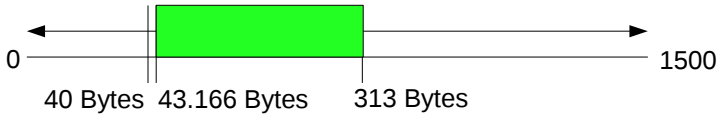
$$minimumaverage = \frac{40B(syn) + 40B(ack) + 59B(query) + 40B(ack) + 40B(fin, ack) + 40B(ack)}{6} \tag{A.1}$$

The outcome of equation A.1 is a absolute minimum average of ≈43.167

For the upper side of the traffic it can be calculated that the largest packet without any option fields can be 313 bytes, however the added option field to either IP or TCP will increase the packet size by a multitude of four. The packets of the three way handshake and the connection



(a) UDP



(b) TCP

Figure A.3: Packet size DNS distributions

teardown will reduce the average number of packets per flow for a single TCP connection, however because a flow could exist with only one packet it is assumed that it is better to use the 313 bytes for a packet as a threshold. It is assumed that impact of the extra bytes for the option fields is limited.

In figure A.3(a) the distribution domain of the average TCP packet size can be seen.

B

Standard deviation

In section 5.2.1 on page 42 the use of standard deviation for finding attacks was described. In this section the effectiveness of this method will be validated. To analyze the effectiveness of this method as much data as possible about the server was gathered, so that it could be definitively concluded that this host was under attack.

For the validation of the standard deviation method the Netflow dataset that was captured in August 2007 was used. The calculated standard deviation was validated with other quantitative measures in the same dataset.

	Average. ppf	Std. deviation	Duration sec.	Peak packets	Total octets	Avr. KiB/s	Peak KiB/s	avr. packet size	requests	responses	IPs
1	874996.8635	120088.51	57573.376	1104195	35983378286	610.35	688.31	46.0000	782247196	0	5
2	135.4876	527.19	204424.256	3234	162841095	0.78	7.53	70.5458	2308302	2305583	38
3	117.4511	491.18	204441.408	3251	187973306	0.90	14.01	81.3066	2311907	2308985	42
4	570.3333	384.13	1725.120	1006	20468981	11.59	33.38	920.2437	22243	13921	1
5	49.7439	70.87	125862.1592	683	74165515	0.58	15.12	91.1449	813710	813553	3
6	4.4362	62.31	351236.264	9766	176636319	0.49	105.64	73.3716	2407422	2406755	8839
7	64.0000	62	7.936	126	18432	2.27	4.71	72.0000	256	0	1
8	50.5000	49.5	28.352	100	7317	0.25	0.25	72.4455	101	26	1
9	2.5833	41.33	6221718.820	65536	4907106138	0.77	210.25	103.3563	47477581	42618822	185724
10	2.1457	36.26	1116546.568	3271	994879219	0.87	145.36	122.3017	8134633	7247008	85560
11	9.6955	33.95	179503.616	762	65104236	0.35	20.51	111.2237	585345	189438	16
12	21.4736	33.71	109371.520	288	34440839	0.30	23.29	85.1131	404648	398713	5
13	7.9787	32.53	259630.464	610	70124981	0.26	29.75	72.3415	969361	969358	42
14	9.1122	32.06	141948.608	762	51081472	0.35	18.78	110.8053	461002	67635	16
15	27.4286	28.95	68.736	78	13362	0.19	0.64	69.5938	192	49	4
16	39.6667	27.34	0.628	59	4884	7.59	8.28	41.0420	119	0	1
17	38.3333	26.40	0.004	57	4891	1194.09	542.72	42.5304	115	0	1
18	38.3333	26.40	0.532	57	5290	9.71	10.00	46.0000	115	0	1
19	38.3333	26.40	0.540	57	4720	8.54	10.00	41.0435	115	0	1
20	38.3333	26.40	0.540	57	4720	8.54	10.00	41.0435	115	0	1
21	37.6667	25.93	0.004	56	5198	1269.04	628.91	46.0000	113	0	1
22	37.6667	25.93	0.548	56	4862	8.66	9.83	43.0265	113	0	1
23	3.1704	23.95	483.968	321	998308	2.01	12.13	74.6175	13379	13379	17
24	3.0069	22.73	389.504	316	814715	2.04	4.96	74.8269	10888	10888	17
25	10.2087	19.88	557.956	142	163588	0.29	13.18	63.0883	2593	2591	13
26	6.8824	19.51	1925.248	201	745459	0.38	10.16	92.3398	8073	8073	15
27	25.7143	19.08	88.704	46	11734	0.13	0.15	65.1889	180	0	2
28	4.4098	17.80	80.960	218	99221	1.20	3.22	73.7703	1345	0	3
29	3.5103	17.39	68996.864	1953	14865670	0.21	5.22	81.8414	181640	181400	32
30	8.6079	17.26	91068.352	243	29975847	0.32	13.11	90.2653	332086	332047	6

Table B.1: Statistics of the top 30 hosts

In table B.1 shows summarized information about the top 30 DNS servers from the standard deviation list. The first thing that is noticeable is that the average throughput for most of these servers is very low. This is not the real average throughput handled by the server, but was calculated using the following formula:

$$average_{throughput} = \sum_{i=1}^{N} octets / \sum_{i=1}^{N} duration \tag{B.1}$$

There is however a flaw in the use of formula B.1. The duration as saved by the Netflow exporter does not represent the actual duration of the transfer. It can be that a flow has duration zero or a duration equal to the Netflow timeout period. Because of this the calculated throughput is only indicative. The deviation from the actual throughput is unknown.

In table B.1 the marked measures indicate that something might be wrong, besides the standard deviation. A closer look at the traffic of the above 30 hosts revealed that 18 of the 30 hosts on this list show signs of probable attacks. They are indicate with the colorized host number in the table. The 12 host that could not validated, show high fluctuations in the number of packets per flow. Not all of these twelve hosts could be completely cleared of being a target of a possible attack. The problem is that the Netflow data itself doesn't give enough information to give a clear answer.

A value that stands out in this validation is the lack of responses of some of the hosts. If the attacker was performing a port scan one could expect only a low amount of packets, which would give only a small standard deviation. The hosts in this list however show hundreds of packets towards it, and have a large standard deviation.

Another observation is that there are some strange average packet size values in table B.1. A packet to the DNS port should in all cases be larger than 44 bytes for UDP traffic (43.167 bytes TCP packet) (see appendix A), so any value indicating an average value lower than this could be considered hostile. Although the DNS protocol has some extension to allow large packets, it is peculiar to have a very large average packet size.

The balance between request and responses can be indicative of an attack. A clear assumption is that within every connection every request that is made should get a response. It is however possible that packets get lost and because of that get retransmitted. Also the balance of packets for TCP and UDP traffic are different, because in TCP a mechanism like cumulative acknowledgment in which an acknowledgment is sent after a timeout period in which other packets can also be received. This makes that there might be less responses than requests.

As stated in section 5.2.1 there is an inherent problem with using the standard deviation as a value for finding attacks. This problem had significant influence on the values of standard deviation that were found in this list. Of the hosts in this list twelve have an increased standard deviation due to lower than average values. Of these hosts eleven seem to have clear signs of attacks, which is about halve of the validated hosts. It seems that this might not be as big a problem as expected.

It has become clear that the standard deviation alone can indicate possible attacks, but cannot do so with 100 % accuracy. To increase the effectiveness of the standard deviation method other measures are needed.

Bibliography

[1] Trends in denial of service attack technology. *CERT*, July 2008. URL http://www.cert.org/archive/pdf/DoS_trends.pdf.

[2] P Albitz and C Liu. *DNS and BIND fourth edition, 10.2. DNS Dynamic Update*. O'Reilly & Associates, 2002. URL http://www.unix.com.ua/orelly/networking_2ndEd/dns/ch10_02.htm.

[3] A. Art Manion and J.A. Rafail. Buffer overflows in multiple dns resolver libraries. *CERT*, June 2002. URL http://www.cert.org/advisories/CA-2002-19.html.

[4] Sir Francis Bacon, 1597. URL http://www.quotationspage.com/quote/2060.html.

[5] S. Bellovin. Defending against sequence number attacks. RFC 1948, Internet Engineering Task Force, May 1996. URL http://www.rfc-editor.org/rfc/rfc1948.txt.

[6] D.J. Bernstein. The dns_random library interface of djbdns. 2000. URL http://cr.yp.to/djbdns/dns_random.html.

[7] P. Biondi. Scapy packet manipulator. URL http://www.secdev.org/projects/scapy/.

[8] J. Boyd. Text of matasano's article on detais of kaminsky's dns attack boydjd's picture. July 2008. URL http://www.jbip.net/content/text-mantasanos-article-detais-kaminskys-dns-attack.

[9] C.J. Brandhorst and A. Pras. Dns: a statistical analysis of name server traffic at local network-to-internet connections. *Electrical Engineering, Mathematics and Compuer Science, University fo Twente*, January 2005. URL http://www.simpleweb.org/nm/research/results/publications/pras/2005-Eunice-Brandhorst.pdf.

[10] C. Brenton. Egress filtering faq. *SANS institute*, January 2007.

[11] N. Brownlee, K.C. Claffy, and E. Nemeth. Dns measurements at a root server. *Global Telecommunications Conference, 2001. GLOBECOM '01. IEEE*, 3:1672–1676 vol.3, 2001. doi: 10.1109/GLOCOM.2001.965864. URL http://ieeexplore.ieee.org/iel5/7633/20834/00965864.pdf?arnumber=965864.

[12] P. Bcher, T. Holz, M. Ktter, and G. Wicherski. Know your enemy: Tracking botnets. *The Honeynet Project & Research Alliance*, March 2005. URL http://www.honeynet.org/papers/bots/.

[13] CERT. Computer emergency response team (cert) homepage, October 2008. URL `http://www.cert.org/`.

[14] Cisco. *Cisco IOS NetFlow*, April 2008. URL `http://www.cisco.com/en/US/products/ps6601/products_ios_protocol_group_home.html`.

[15] G Combs. Wireshark packetsniffer. September 2008. URL `http://www.wireshark.org/`.

[16] P.T. De Boer, D.P. Kroese, S. Mannor, and R. Y. Rubinstein. A tutorial on the cross-entropy method. *Annals of Operations Research*, 134, 2004. URL `http://iew3.technion.ac.il/CE/files/papers/CEtutfinal.pdf`.

[17] R. Droms. Dynamic host configuration protocol. RFC 2131, Internet Engineering Task Force, March 1997. URL `http://www.rfc-editor.org/rfc/rfc2131.txt`.

[18] MSN Encarta. Definition entropy. December 2008. URL `http://encarta.msn.com/encnet/features/dictionary/DictionaryResults.aspx?refid=1861608694`.

[19] P. Ferguson and D. Senie. Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing. RFC 2827, Internet Engineering Task Force, May 2000. URL `http://www.rfc-editor.org/rfc/rfc2827.txt`.

[20] S. Gibson. The strange tale of the denial of service attacks against grc.com. *Gibson Research Corporation*, March 2002. URL `http://www.grc.com/dos/grcdos.htm`.

[21] K. Houle. Denial of service attacks using nameservers. *CERT*, April 2000. URL `http://www.cert.org/incident_notes/IN-2000-04.html`.

[22] IANA. Domain name system (dns) parameters. July 2008. URL `http://www.iana.org/assignments/dns-parameters`.

[23] Internet Assigned Numbers Authority (IANA). Port numbers. September 2008. URL `http://www.iana.org/assignments/port-numbers`.

[24] Internet Assigned Numbers Authority (IANA). Top level domains. September 2008. URL `http://data.iana.org/TLD/tlds-alpha-by-domain.txt`.

[25] ICANN. Icann homepage. December 2008. URL `http://www.icann.org/`.

[26] C. Irving. The achilles heal of dns. *SANS Institute*, October 2003. URL `http://www.sans.org/reading_room/whitepapers/dns/565.php`.

[27] Simon Josefsson. The base16, base32, and base64 data encodings. Rfc, Internet Engineering Task Force, October 2006. URL `http://www.rfc-editor.org/rfc/rfc4648.txt`.

[28] G. Kambourakis, T. Moschos, D. Geneiatakis, and S. Gritzalis. A fair solution to dns amplification attacks. *Digital Forensics and Incident Analysis, 2007. WDFIA 2007. Second International Workshop on*, pages 38–47, Aug. 2007. doi: 10.1109/WDFIA.2007.4299371. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4299371&isnumber=4299360`.

[29] A. Karasaridis, K. Meier-Hellstern, and D. Hoeflin. Nis04-2: Detection of dns anomalies using flow data analysis. *Global Telecommunications Conference, 2006. GLOBECOM '06. IEEE*, pages 1–6, Nov. 2006. ISSN 1930-529X. doi: 10.1109/GLOCOM.2006.280. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=4150910&isnumber=4150630`.

[30] J. Kinable. Detection of network scan attacks using ow data. *Electrical Engineering, Mathematics and Compuer Science, University fo Twente*, June 2008. URL http://dacs.ewi.utwente.nl/assignments/completed/bachelor/reports/2008-kinable.pdf.

[31] John C Klensin. National and local characters for dns top level domain (tld) names. RFC 4185, Internet Engineering Task Force, October 2005. URL http://www.rfc-editor.org/rfc/rfc4185.txt.

[32] D. Kramer. Definition: Buffer overflow. *searchsecurity.com*, June 2007. URL http://searchsecurity.techtarget.com/sDefinition/0,,sid14_gci549024,00.html.

[33] C Kreibich, A. Warfield, J. Crowcroft, S. Hand, and I. Pratt. Using packet symmetry to curtail malicious traffic. *University of Cambridge Computer Laboratory*, August 2005. URL http://www.sigcomm.org/HotNets-IV/papers/kreibich.pdf.

[34] S Kullback. *Information Theory and statistics*. John Wiley & Sons, Inc, 1959.

[35] S. Kullback and R.A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics 22 (1951) (1), pp. 7986*, 1951.

[36] J.P. Lanza, C. Cohen, R. Danyliw, I Finlay, S. Hernan, and Q.R. Peyton. Cert advisory ca-2001-02 multiple vulnerabilities in bind. *CERT*, August 2001. URL http://www.cert.org/advisories/CA-2001-02.html.

[37] C.B. Lee, C. Roedel, and E. Silenok. Detection and characterization of port scan attacks. *Department of Computer Science & Engineering University of California, San Diego*, March 2003. URL http://www.cse.ucsd.edu/users/clbailey/PortScans.pdf.

[38] J. Levine. Dns blacklists and whitelists draft-irtf-asrg-dnsbl-08. Technical report, Internet Engineering Task Force, July 2008. URL http://tools.ietf.org/html/draft-irtf-asrg-dnsbl-08.

[39] R.W. Maple. [esa-20021114-029] bind buffer overflow, dos attacks. *EnGarde Secure Linux*, November 2002. URL http://www.derkeiler.com/Mailing-Lists/securityfocus/bugtraq/2002-11/0224.html.

[40] L. MartinGarcia. Tcpdump/libpcap public repository, October 2008. URL http://www.tcpdump.org/.

[41] P. Mockapetris. Domain names - concepts and facilities. RFC 1034, Internet Engineering Task Force, November 1987. URL http://www.rfc-editor.org/rfc/rfc1034.txt.

[42] P. Mockapetris. Domain names - implementation and specification. RFC 1035, Internet Engineering Task Force, November 1987. URL http://www.rfc-editor.org/rfc/rfc1035.txt.

[43] D. Moore, G.M. Voelker, and S. Savage. Inferring internet denial-of-service activity. *Department of Computer Science and Engineering, University of California*, pages 9–22, May 2001. URL http://www.caida.org/outreach/papers/2001/BackScatter/usenixsecurity01.pdf.

[44] M. Ohta. Incremental zone transfer in DNS. RFC 1995, Internet Engineering Task Force, August 1996. URL http://www.rfc-editor.org/rfc/rfc1995.txt.

[45] V. Paxson. An analysis of using reflectors for distributed denial-of-service attacks. *SIGCOMM Comput. Commun. Rev.*, 31(3):38–47, 2001. ISSN 0146-4833. doi: http://doi.acm.org/10.1145/505659.505664.

[46] J. Plenz. Dnstunnel.de, 2006. URL `http://dnstunnel.de/`.

[47] J. Postel. User datagram protocol. RFC 768, Internet Engineering Task Force, August 1980. URL `http://www.rfc-editor.org/rfc/rfc768.txt`.

[48] J. Postel. Obsoletes RFCs: 776, 770, 762, 758, 755, 750, 739, 604, 503, 433, 349 obsoletes IENs: 127, 117, 93. RFC 790, Internet Engineering Task Force, September 1981. URL `http://www.rfc-editor.org/rfc/rfc790.txt`.

[49] K. Ramakrishnan, S. Floyd, and D. Black. The addition of explicit congestion notification (ECN) to IP. RFC 3168, Internet Engineering Task Force, September 2001. URL `http://www.rfc-editor.org/rfc/rfc3168.txt`.

[50] J. Reynolds and J. Postel. Assigned numbers. RFC 1700, Internet Engineering Task Force, October 1994. URL `http://www.rfc-editor.org/rfc/rfc1700.txt`.

[51] D.S. Sax. Dns spoofing (malicious cache poisoning). *SANS Institute*, May 2003. URL `http://www.giac.org/certified_professionals/practicals/gsec/0189.php`.

[52] S.M. Specht and R.B. Lee. Distributed denial of service: Taxonomies of attacks, tools and countermeasures. *Proceedings of the 17th International Conference on Parallel and Distributed Computing Systems, 2004 International Workshop on Security in Parallel and Distributed Systems*, pages 543–550, September 2004. URL `http://palms.ee.princeton.edu/PALMSopen/DDoS%20Final%20PDCS%20Paper.pdf`.

[53] J. Stewart. Dns cache poisoning - the next generation. *SecureWorks*, April 2003. URL `http://www.lurhq.com/dnscache.pdf`.

[54] Wikipedia the free encyclopedia. Pseudorandom number generator, October 2008. URL `http://en.wikipedia.org/wiki/Pseudorandom_number_generator`.

[55] Berkeley University of California. Bind dns software. September 2008. URL `http://www.isc.org/index.pl?/sw/bind/index.php`.

[56] D van der Sanden. Detecting udp attacks in high speed networks using packet symmetry with only flow data. *Electrical Engineering, Mathematics and Computer Science, University fo Twente*, June 2008. URL `http://dacs.ewi.utwente.nl/assignments/completed/bachelor/reports/2008-IOO-Daan.pdf`.

[57] Heise Zeitschriften Verlag. Icann and iana domains hijacked. URL `http://www.heise.de/english/newsticker/news/110279`.

[58] P. Vixie. Extension mechanisms for DNS (EDNS0). RFC 2671, Internet Engineering Task Force, August 1999. URL `http://www.rfc-editor.org/rfc/rfc2671.txt`.

[59] P. Vixie. Extensions to dns (edns1). Technical report, Internet Engineering Task Force, August 2002. URL `http://www.ops.ietf.org/lists/namedroppers/namedroppers.2002/msg01384.html`.

[60] Paul Vixie, Sue Thomson, Y. Rekhter, and Jim Bound. Dynamic updates in the domain name system (DNS UPDATE). RFC 2136, Internet Engineering Task Force, April 1997. URL `http://www.rfc-editor.org/rfc/rfc2136.txt`.

[61] D. Wessels and Fomenkov M. Wow, that's a lot of packets. *PAM, 2003*, 2003. URL http://www.caida.org/outreach/papers/2003/dnspackets/wessels-pam2003.pdf.

[62] the free encyclopedia Wikipedia. Type i and type ii errors. December 2008. URL http://en.wikipedia.org/wiki/Type_I_error.

[63] M. Zalewski. Strange attractors and tcp/ip sequence number analysis. 2001. URL http://lcamtuf.coredump.cx/oldtcp/tcpseq.html.