Open Recursive Nameservers

What's the problem with that?

Students:
P. van Abswoude
P. Tavenier

Supervising teachers:
J.P. Velders
K. Koymans

System and Network Engineering
University of Amsterdam

Final Version

February 5, 2007

Abstract

The transition of 2005 into 2006 was marked by several high-impact Distributed Denial of Service (DDoS) attacks, totaling traffic exceeding the 10Gb/s mark, where machines were involved in the attacking role which were ultimately abused because they offered (un)intentionally DNS services to everyone and their toaster on the Internet.

During the month January of the year 2007 we conducted research concerning Open Recursive Nameservers. We conducted both theoretical as well as practical research around Open Recursive Namservers.

We experienced that approximately 46% of the nameservers is Caching Open Recursive. We came to these results by doing research on the .int and .edu zonefiles.

We discovered that DNSSEC can be used to create a 4 times bigger DNS UDP answer – the answer increased from 512 bytes to 2048 bytes.

We also conducted research around the bandwidth that is needed to execute a DNS DDoS attack. With practical tests we come to the conclusion that about 3% of the bandwidth of the target you're going to attack is needed.

Acknowledgments

First of all we would like to thank J.P. Velders of the University of Amsterdam for his technical support during our research period. We also would like to thank K. Koymans of the University of Amsterdam for his support and suggestions during our project.

Last but not least we would like to thank the University of Amsterdam for the hardware we could use for the duration of our research.

Contents

Definitions

Client: Typically a “stub-resolver” on an end-user’s computer.

Resolver: Preferred nameserver performing recursive service for clients.

Query/Question: A DNS query send out by a client or resolver, typically in a UDP packet.

Reply/Answer: The DNS answer send back by a nameserver, typically in a UDP packet.

Attacker: Malicious third party.

Spoof: The activity of changing the source IP-address in an IP packet.

Target/Victim: Host that is being subjected to a DNS DDoS attack.

Administrator: When talking about administrators we refer to system- and/or network administrators who are responsible for a nameserver.

Firewall: When talking about a firewall we mean both a software as a hardware firewall solution.

In the whole document, the words CORN and ORN will be frequently used. CORN stands for Caching Open Recursive Nameserver and ORN stands for Open Recursive Nameserver.

Chapter 1

Introduction

The transition of 2005 into 2006 was marked by several high-impact Distributed Denial of Service (DDoS) attacks, totaling traffic exceeding the 10Gb/s mark, where machines were involved in the attacking role which were ultimately abused because they offered (un)intentionally DNS services to everyone and their toaster on the Internet.

In this research project theoretical and practical research has been done on DNS DDoS attacks concerning Open Recursive Name Servers. Our main goal is to emphasize the dangers of (unconsciously) running a Open Recursive Name Server. We also delivered guide lines for System Administrators, Network Engineers and IT-Architects how to secure their DNS servers.

1.1 Researched sections

The following subsections describe the practical tests and theoretical research we conducted during the research period.

1.1.1 Test 1 – Reconnaissance work

In this section we summarized the reconnaissance work that is done by an potential attacker. This contains creating a list of Open Recursive Name Servers and what the rate of return[1] is for an attacker. We discussed multiple attack tactics for abusing open recursive DNS servers to conduct a DNS DDoS attack. We also performed statistical analysis on (Caching) Open Recursive Nameservers.

[1]A comparison of the profit on an attack to the amount of effort.

1.1.2 Test 2 – Defend tactics

In this test we created our own Open Recursive Name Server (BIND9) and tested several possibilities to prevent abuse. This section also describes possible solutions that could be implemented on the DNS server itself as well as on connected firewalls and/or routers. We created this section by summarizing which factors played what role in facilitating these attacks.

Chapter 2

What is a (Caching) Open Recursive Nameserver?

The Domain Name System (otherwise known as DNS) is one of the most used protocols used on the Internet. The protocol exists since 1983 (RFC 882[1] and 883[2]) and has been updated around 1987 (RFC 1034[3] and 1035[4]). Since then, there have been no other "major" updates [1] to the protocol so the protocol we know as DNS has been around for 20 years – and we guess that many years will follow.

Since 1987 there have been various DNS server implementations. BIND is the first (open source) UNIX implementation. Since then a lot of other companies/projects came with there own DNS implementation. A few examples are Microsoft DNS[5], PowerDNS[6], MaraDNS[7], and many more. We're sure when you search the Internet you will be dazzled by the different kinds of DNS servers.

A DNS server can have various roles. The following list will give the most common roles of a DNS server:

- Master (primary) – primary authoritative server for a zone.
- Slave (secondary) – secondary authoritative server for a zone.
- Stub (limited secondary) – secondary authoritative server for a part of a zone.
- Stealth (secondary that is not listed) – secondary authoritative which is not registered in a zone.

[1] There have been updates of DNS in the form of DNS extensions designed by the IETF. More information about DNS extensions can be found on `http://www.ietf.org/html.charters/dnsext-charter.html`.

- Caching-only (never authoritative) – a DNS server that answers from cache.
- Forward-only (using “forwarders”) – Forwards the query to another recursive DNS server.

From the beginning of DNS a function called ‘Recursion’ has been implemented in the various server implementations. Recursion makes it possible for a nameserver to look up the asked domain because the preferred nameserver follows referrals itself. The best way to explain this is with figure 2.1 and a concrete step-by-step example[2].

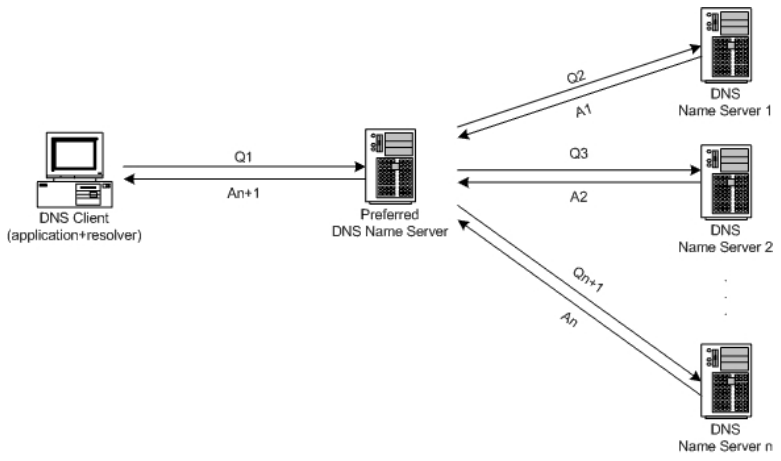


Figure 2.1: Recursion in DNS

1. Q1 – The client asks his preferred nameserver for “www.os3.nl.”.
2. Q2 – The preferred server doesn’t know where “www.os3.nl.” is and query the *root* (.) nameserver.
3. A1 – The *root* nameserver answers that more information about “www.os3.nl.” is available at the *.nl.* nameserver.
4. Q3 – The preferred server than query the *.nl.* nameserver for “www.os3.nl.”.
5. A2 – The *.nl.* nameserver answers that more information about “www.os3.nl.” is available at *ns1.os3.nl.*
6. Q4 – The preferred server than asks the *ns1.os3.nl* nameserver for “www.os3.nl.”.

[2]Picture originally from `http://www.dei.isep.ipp.pt/~nsilva/ensino/asi1/asi1%202005-2006/images/dns_recursive_lookup.gif`

7. A3 – The *ns1.os3.nl.* answers that “www.os3.nl.” can be found at 145.92.26.20.

8. A4 – The preferred servers answers the query from the client with *145.92.26.20*.

In this example, the preferred DNS server is a “Recursive Nameserver” for the clients inside its network, because the preferred DNS server resolves the queries from the clients.

2.1 Open Recursive Nameservers

In the example above the client and preferred nameserver are in the same network and the preferred nameserver is a recursive nameserver. This is not a problem, because without a recursive preferred nameserver no domain name could ever be resolved. The problem lies in the word **open**. An ORN is a nameserver that will resolve queries from all clients, also the ones that are located outside its own domain/network.

Normally a nameserver only serves the information about one particular domain to clients from the outside world. This means that the authoritative nameserver of os3.nl. will only answer to queries concerning *.os3.nl. (the “*” represent all sub-domains of os3.nl.). However an *open* recursive nameserver will also answer queries located outside its own domain from clients that are located outside its own network. That means that if the nameserver of os3 gets a query to resolve “www.uva.nl.” it will answer the query with “145.18.11.202”. If the nameserver of os3.nl. doesn’t know where it is, it will search for an answer for you and will give the answer back to you. And that’s not really its task – meaning that the nameserver should only serve its own domain to clients of the outside world!

2.2 Caching Open Recursive Nameservers

A query is normally a question for the retrieval of a Resource Record (RR) from a zone. RRs contain various properties. First of all a RR is part of a so called ‘zone’. Zones are described in zonefiles[3] – an administration file for a nameserver. A zonefile contains the records of a specified zone or domain. A zone has various properties of which the TTL – the Time To Live for a record in a zone – is the most important in our research.

A Caching Open Recursive Nameserver (further: CORN), will not only answer to a query outside its own domain, it will also cache – or store the

[3]This is an specific BIND9 option

answer for the Time To Live (TTL) that is specified for that record. This is done for a simple reason explained in this example.

If the nameserver gets for example the query www.os3.nl., and it doesn't know where it is, it will search for the answer and cache it for a duration in seconds defined by the TTL, which is defined in the zone file of os3.nl. If another client asks for www.os3.nl. before the TTL is expired, it will answer from cache so that the nameserver doesn't spill bandwidth by querying again.

2.3 Are CORNs bad?

CORNs itself aren't necessarily bad. It's really nice of all the network administrators who let their nameserver accept queries from outsiders and resolve them for you. I.e. A manager of a company who is traveling to other countries just wants to use the Internet without any trouble so he uses the nameserver of the company itself. The administrator at the other side of the world wants to have a nice sleep, because he doesn't want to solve the Internet problems of the manager day and night. The administrator runs a ORN so the manager can use the DNS server of the company. But, unfortunately there are also miscreants who could abuse a (C)ORN for their own miscreant-things.

Let's say you are such a miscreant and you want to annoy someone. You could make a list of CORNs, let them cache a query you want (because they are open recursive and they cache), and send a bogus query to the nameserver containing the IP-address of the victim. This means that the CORNs will answer to that IP and the Internet connection of the victim will be flooded with UDP traffic so that it can't function normally. This is no (big) problem for a small company, but imagine the financial damage for online stores like Amazon, E-bay or big-time companies like Microsoft.

Chapter 3

Reconnaissance work

3.1 Introduction

Before an attacker can attack a target, he needs a list of (C)ORNs. In section 3.2 we describe how you can get a list of (C)ORNs. How an attacker attack his target will be described in section 4.5.1.

3.2 Find/Make a list of name servers

Before an Attacker can start his attack he need a list of ORNs to abuse. Here is an example how to get a list with ORNs:

1. To start, you need a list of domain names.
2. Query these domain names and filter which nameserver they use. Now you have a list of nameservers. Some of these lists are even as download available.
3. Query at these nameservers a domain name that's in your own domain. If you get an answer and a query in your log, than it is an ORNs.
4. Query the same domain name at these nameserver. If you get an answer and by no query in your log, than it is an caching ORNs.
5. Sort this list and you have a list of CORNs!

You can imagine that it isn't that hard to create a list of CORNs. Sometimes you can skip the first step when you already have list of nameservers. For Example, the .edu and the .int zone files are free available at the Internic website. And if you register yourself at VeriSign you can even get a list of the other domains, like .com, .net, etc.

We did research on ORNs ourself with the .edu and the .int nameserver list. The results of this research are described in section 4.3.

3.3 Get control over bot network

When the attacker is planning to attack a big company he may need to create a lot of bandwidth. Since our research results (section 4.3) indicate that we need about 3% of that bandwidth, the total upload bandwidth for attacking a 10Gb/s line must be around 300Mb/s. With the current state of the home-users DSL connections (which is about 256Kb/s - 1024Kb/s upload) this can be archived with about 300-1000 bots on such DSL connections.

How a bot network is created and how attackers get control over these networks is out of the scope of the project. Vint Cerf mentioned January 25th 2007 at the World Economic Forum in Davos[10] that probably 100-150 million computers are part of a bot network. So you can imagine that it isn't difficult to get a big bot network if you know the right (read: wrong) people.

3.4 What is the profit for an attacker

An attacker won't attack anything if the chance to get caught is high and the profit is low. You can see this as a Return on Investment (ROI)[8]. Some advantages and disadvantages are for an attacker are:

- Advantages
  - Easy to find CORNs
  - Bandwidth profit is huge (See section 4.3)
- Disadvantages
  - Creating a bot network
  - remain undetected

An attacker can have different reasons to perform an attack, for example:

- Extortion (In exchange of a ransom, he will stop the attacks).
- Disagreement (Financial reasons with ex-boss; Political reasons).
- Show-off (Show what he is capable of).
- Just for fun (See if it's possible to attack a big target).
- Annoy (It's know that script kiddies use it on IRC to kick each other from the Internet).
- Gain profit with online gaming[9].

Chapter 4

Practical Research

4.1 Introduction

This part of the document will explain the practical research we did during our research period. We performed two sets of tests. In the first tests we made a list of CORNs and in the second tests we executed an DNS DDoS attack in our local network to test how we could abuse CORNs.

4.2 Test Environments

For the first tests we used two servers with Ubuntu 6.06 (Dapper) installed and the kernel version we used was 2.6.15-27. On one server we also installed the standard installation of BIND 9.3.2 via apt, where the DNSSEC extension was enabled by default.

In BIND we configured a domain with a wildcard. This made it possible to sent out a query with the IP in front of the wildcard (see section 4.5.2). In the query.log from BIND we could check if the IP in the query matched with the IP we queried. If the IP's where the same, we know that is was an ORN, but if the IP's differed, it could be a forwarding nameserver. We go further into this subject in section 4.4.

In the following steps we will explain what happens in figure 4.1.

- Q1 is a query "<ip>.test.os3.nl IN TXT".
- Q2 is the same query sent to NS if the nameserver on the Internet is open recursive.
- the wildcard on NS makes it possible to reply on the query.
- A1 is the anwser of the query to the ORN on the Internet. It's possible that the nameserver cashes the answer.
- A2 is the anwser back to Q.

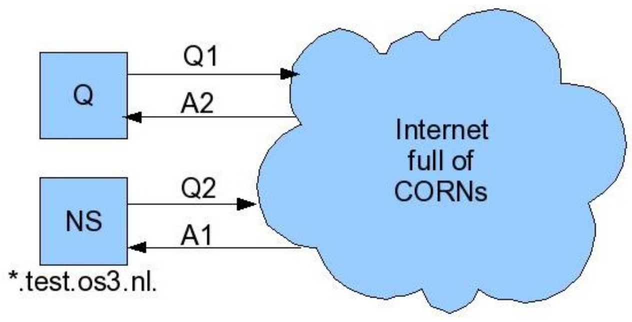


Figure 4.1: The search for CORNs test environment

For the second test we had a victim with Ubuntu 6.10 (Edgy) installed and kernel 2.6.17-10. We used the same nameservers from the first test. More details about this attack technique using TXT records is discussed in section 4.5.2.

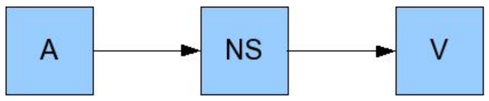


Figure 4.2: The DNS DDoS test environment

In figure 4.2 the test environment is displayed. In the following steps we will explain how used these three computers to perform an attack.

- A (the attacker) sends out a query for a TXT record to NS (the nameserver). The IP-packet is spoofed (using Perl), which means in our case that the sender IP in changed into V (the victim) instead of A.
- NS receives the query and sends the TXT record, that is made as big as possible (see section 4.5.2), to V.
- V receives the UDP IP-packet with the answer.

If you perform these step with more “A” computers, which you have if you have the control over an botnetwork, you have a big distributed attack.

4.3 Zonefiles research statistics

We did research on the zonefiles of the .edu and .int domains. We downloaded these files form the Internic ftp server. The serial numbers of the files we used were:

- edu.zone = 2006042800

- int.zone = 2007011100

In the sections 4.3.1 and 4.3.2 we distinguished nameservers that are **inside** the zone from nameservers that are **outside** the zone.

In our tests we encountered 4 kinds of servers:

- Open nameservers– The open recursive nameservers.
- Forwarders – nameservers that forward a query to another nameserver who queried and answered.
- Closed nameservers – Nameservers that are not an ORN.
- Connection Time-out nameservers – nameservers that weren't reachable in any way.

Because the connection time-out servers don't reply on *any* query, you could wonder if those nameservers are even online. It could be that there is an error in the zonefile or that the administrators brought the nameserver offline. Therefore we calculated two statistics; one with the connection time-out servers **included** with the total amound of nameservers, and one with the connection time-out servers *deducted* from the total amound of nameservers. We did this because the second group doesn't respond to anything so we deducted the timed-out servers from the total amound nameservers.

4.3.1 int zonefile statistics

In table 4.1 the .int zonefile has 12 ORNs and 9 forwarders. that means 21 nameservers which are open recursive. With the second query, within the TTL value, all the 21 ORNs answered the query but didn't query our nameserver. This means that they are cashing. 100% of the ORNs are CORNs. 8 of the nameservers in table 4.1 had a time-out, that means that

—	amount of servers	percentage
Total nameservers	59	100%
Open Nameservers	12	20%
Forwarders	9	15%
Closed nameservers	30	51%
Connection Time-out	8	14%

Table 4.1: nameservers of the .int zonefile inside the .int domain

there are 51 running nameservers left. 21 of the total 51 nameservers are CORNs, that's 41%. Including the timed-out nameservers it's still 36%.

The nameservers in table 4.2 are outside the .int domain (that means they are in the .net, .com, etc. domain). There where 68 ORNs (including the forwarders). By the second query within the TTL value only sent to the

ORNs, 3 nameservers queried our nameserver again. That means they're not cashing but the other 96% of the ORNs are CORNs.

—	amount of servers	percentage
Total nameservers	203	100%
Open nameservers	49	24%
Forwarders	19	9%
Closed nameservers	127	63%
Connection Time-out	8	4%

Table 4.2: nameservers of .int zonefile outside the .int domain

8 of the nameservers in table 4.2 had a time-out, that means that there are 195 running nameservers left. 65 (68-3) of the total 195 nameservers are CORNs, that's 33%. Including the timed-out nameservers it's still 32%.

4.3.2 edu zonefile statistics

The .edu zonefile in table 4.3 has a total of 2163 ORNs (including the forwarders). By the second query within the TTL value only sent to the ORNs, 21 nameservers queried our nameserver again. So 99% of the ORNs are CORNs. 931 of the nameservers in table 4.3 had a time-out, that means

—	amount of servers	percentage
Total nameservers	4264	100%
Open nameservers	1407	33%
Forwarders	756	18%
Closed nameserver	1170	27%
Connection Time-out	931	22%

Table 4.3: nameservers of .edu zonefile outside de .edu domain

that there are 3333 running nameservers left. 2142 (2163-21) of the total 3333 nameservers are CORNs, that's 64%. Including the timed-out nameservers it's still 50%.

The nameservers in table 4.4 are outside the .edu domain. There were 2302 ORNs (including the forwarders). By the second query within the TTL value only sent to the ORNs, 129 nameservers queried our nameserver again. That means 94% (2173 nameservers) of the ORNs are CORNs.

572 of the nameservers in table 4.4 had a time-out, that means that there are 4552 running nameservers left. 2173 (2302-129) of the total 4552 nameservers are CORNs, that's 47%. Including the timed-out nameservers it's still 42%.

—	amount of servers	percentage
Total nameservers	5124	100%
Open nameservers	1119	22%
Forwarders	1183	23%
Closed nameserver	2250	44%
Connection Time-out	572	11%

Table 4.4: nameservers of .edu zonefile outside the .edu domain

4.3.3 How many CORNs did we find?

In table 4.5 a summarized survey gives us a good impression of the total CORNs. We didn't distinguish ORNs and forwarders, we counted them both as ORNs and if they where cashing as CORNs In table 4.6 the totals

Zonefile	ORNs	CORNs	CORNs (no time-outs)
.int (inside .int domain)	36%	36%	41%
.int (outside .int domain)	33%	32%	33%
.edu (inside .int domain)	51%	50%	64%
.edu (outside .edu domain)	45%	42%	47%

Table 4.5: Global statistics .edu en .int zonefile

of all test are summarized (the abbreviation NS refers to nameservers). 46% of the nameservers in the .edu and .int domain is a CORN. If time-outs are excluded it's even 54%.

Zonefile	NS	NS (without timed-out)	CORNs
.int (inside .int domain)	59	51	21
.int (outside .int domain)	203	195	65
.edu (inside .edu domain)	4264	3333	2142
.edu (outside .edu domain)	5124	4552	2173
totals	9650	8131	4401

Table 4.6: Total numbers zonefiles statistics

Notice that the percentage of timed-out nameservers in higher in the .edu domain. The older serial number (assuming that serial numbers are numbered according RFC 1982[11]) indicates that the zonefile isn't updated for a while. This doesn't mean that we had a to old zonefile, but it could be because there are not many changes in de educational zone. University usual have enough nameservers and it could be that some of the timed-out nameservers are backup servers, which are switched of when they're not needed.

4.4 Problem/advantage with Forwarders

In this research we didn't focus on the fact if the nameserver who resolved the query behind the forwarders are open recursive as well. We counted these "forwarders" as CORNs, because they cache and answer. The percentages of the forwarders are part of the ORNs. If the nameservers behind these forwarders are also open, more (C)ORNs become available to abuse. If we had more time to research this, we could check if these nameservers are open recursive as well.

These forwarders could be a security issue for the administrators who configured them. If an Administrator configured his DNS server correctly and only allows recursion from his own domain. A forwarder in that domain forwards the open recursive queries to the authoritive nameserver. The authoritive nameserver doesn't know if that query is from a client outside his domain because it looks like a client from inside his domain is querying, so the authoritive nameserver resolves the query. So a correctly configured authoritive nameserver could be abused as well.

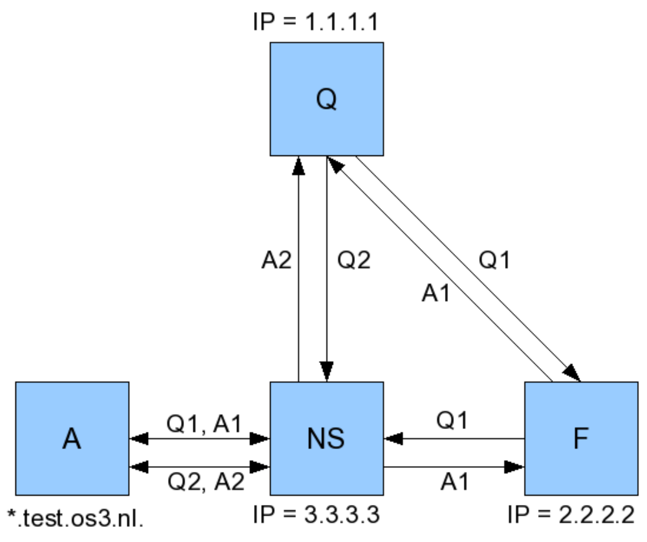


Figure 4.3: Forwarder and nameserver

With a simple test it's possible to check if these nameservers are ORNs. In the picture 4.3 you see:

- Q (Query client) – (IP = 1.1.1.1)
- F (Forwarder) – (IP = 2.2.2.2)
- NS (Nameserver) – (IP = 3.3.3.3)

- A (Nameserver with test data = *test..os3.nl.)
- Q1 (Query from attacker = 2.2.2.2.test.os3.nl.)
- Q2 (Query form attacker = 3.3.3.3.test.os3.nl.)
- A1 & A2 (Answer = IP address)

In the following steps this figure is explaned:

1. Q sends Q1 to F
2. F forwards it to NS, because the answer to Q1 isn't in the cash.
3. NS now resolves the query using A and wil get the answer A1.
4. NS will give the answer A1 back to F, and F replies it to Q.

Q got the answer for his query and in the logfile of A you can see that Q1 (2.2.2.2.test.os3.nl.) is asked by 3.3.3.3. This means there is an forwarder between the Q an A. Now we're able to check if the nameserver who queried A is a ORN.

1. Q sends Q2 *directly* to NS
2. NS now resolves the query using A and wil get the answer A2.
3. NS will give the answer A2 back *directly* to Q.

Q got an answer and in the logfile of A we see that the query 3.3.3.3.test.os3.nl. aksed to NS is queried by NS (3.3.3.3) as well. This means that NS is open recursive as well.

4.5 DNS and UDP/TCP

Most DNS servers work with the User Datagram Protocol (further: UDP). A UDP DNS packet has a maximum size of 512 bytes. This doesn't mean that a answer from a DNS server can't be bigger than 512 bytes. When an answer turns out to be bigger than 512 bytes, most nameservers will generate a so-called 'truncated' reply and will switch to the Transmission Control Protocol (further: TCP). This is done because the client creates a TCP session between client and DNS server, making it possible to send bigger chunks of data – which results in the possibility to give a greater reply.

4.5.1 DNS DDoS and UPD/TCP

The reason why it's important to know the maximum packet size, is because you need UDP with a DNS DDoS attack. We have the following reason for this.

TCP[20] needs to create a session before it can send data. A TCP session is set up between 2 hosts with a 3-way handshake. The client sends a "SYN" to the server. In response, the server replies with a SYN-ACK. Finally the client sends an ACK (usually called SYN-ACK-ACK) back to the server. At this point, both the client and server have received an acknowledgment of the connection.

Every DNS packet has a so called transaction ID[15]. A transaction ID can be used by clients to match requests with responses. When sending a query to a nameserver with UDP, the server will give a response with an UDP packet with the same transaction ID. If the response is bigger than the maximum packet size (512 bytes) the flag "truncated" will be enabled in the UDP packet. Because of the truncated flag, the client will start a TCP session with the server so that all the data will be received. When a UDP packet is being spoofed, the victim will receive a packet with a certain transaction ID and the truncated flag enabled. Because the transaction ID is unknown to the client, there **won't** be a TCP session created thus the CORN can't send its data.

UDP works slightly different. With UDP there is no need for a session. The server simply "pushes" the data to the client. This means in a DNS DDoS attack the victim will always receive the UDP packets but drops them because there is no need for them. This means that his connection is either way filled with UDP packets making the victim's Internet connection non responsive.

There are methods known to influence a TCP handshake but because DNS works primarily with UDP, it is in our opinion easier to abuse UDP than faking a TCP handshake.

4.5.2 TXT records

A standard DNS query size is between 30 and 50 bytes and a 'standard' answer around 200 bytes. When you are able to create an answer of the maximum size of the UDP packet size (512 bytes) you are able to have a profit of 10 times (for each byte that's sent out, you get a 10 byte answer). This is possible by creating a so called TXT record[16]. A TXT record is a record that consists of human-readable characters (ASCII characters) with a sentence, for example who the responsible person for the DNS server is. These day's it's mostly used for the SPF (Sender Policy Framework) which makes it able to reduce spam. The maximum length of a TXT record is 65535 bytes long. The TXT record consists of strings that are 255 bytes surrounded by quotes. An example of an TXT record is:

```
0.test.pvabswoude.practicum.os3.nl. 3600 IN TXT
"This is a test by SNE students of the University of Amsterdam"
"more information is available at www.os3.nl/~ptavenier"
```

With a TXT record you are able to generate an answer of 512 bytes.

4.6 DNSSEC

4.6.1 Introduction

Since the beginning of DNS there have been serious problems with DNS spoofing and DNS cache poisoning[17]. With DNS spoofing, a miscreant tries to give a malicious answer. The definition of DNS cache poisoning from Wikipedia states:

> DNS cache poisoning is a technique that tricks a Domain Name Server (DNS server) into believing it has received authentic information when, in reality, it has not. Once the DNS server has been poisoned, the information is generally cached for a while, spreading the effect of the attack to the users of the server.

4.6.2 What is DNSSEC?

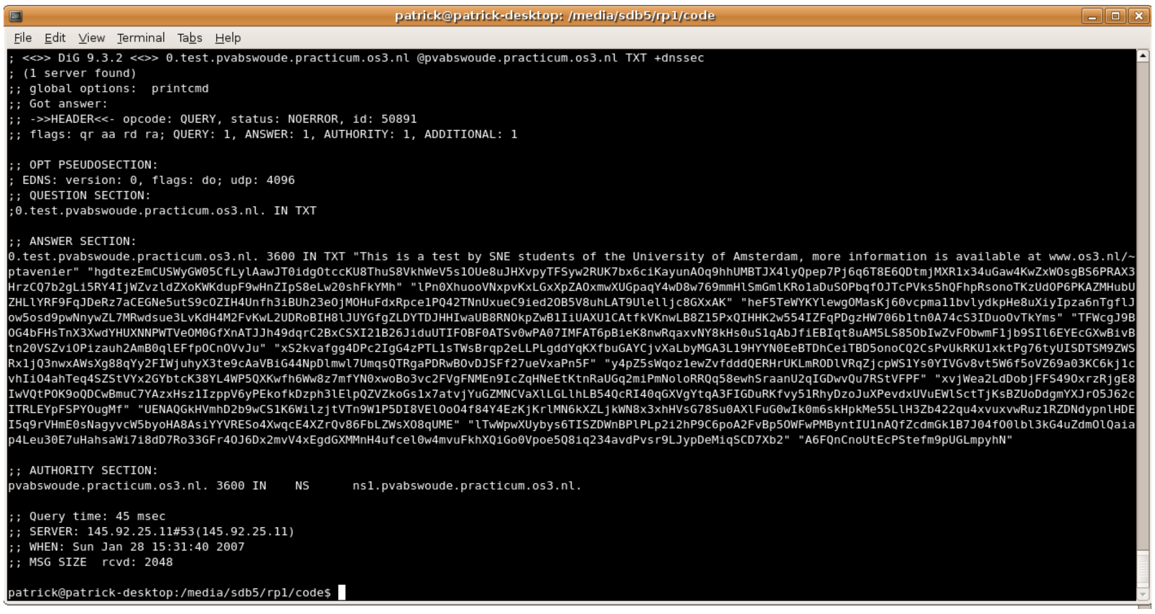
DNSSEC[19] stands for Domain Name System Security Extensions and has been designed by the IETF. The protocol is official since March 2005 and enables DNS clients to:

- Origin authentication of DNS data.
- Data integrity.
- Authenticated denial of existence.

In short, DNSSEC enables DNS clients to authenticate a DNS answer by recursively check if the answer is authentic. This will solve the problems mentioned in section 4.6.1.

4.6.3 DNSSEC and UDP

With the introduction of DNSSEC the problem of spoofing a DNS answer has been made very hard – maybe even impossible. However, there has been created a serious security flaw in UDP. As noted in section 4.5 the maximum UDP packet size is 512 bytes. Because a DNSSEC answer is much bigger than a ‘normal’ DNS answer, they increased the maximum packet size from 512 bytes to 2048 bytes.



```
; <<>> DiG 9.3.2 <<>> 0.test.pvabswoude.practicum.os3.nl @pvabswoude.practicum.os3.nl TXT +dnssec
; (1 server found)
;; global options:  printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 50891
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 1, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags: do; udp: 4096
;; QUESTION SECTION:
;0.test.pvabswoude.practicum.os3.nl. IN TXT

;; ANSWER SECTION:
0.test.pvabswoude.practicum.os3.nl. 3600 IN TXT "This is a test by SNE students of the University of Amsterdam, more information is available at www.os3.nl/~
ptavenier" "hgdtezEmCUSWyGW05CfLylAawJT0idgOtccKU8ThuS8VkhWeV5s1OUe8uJHXvpyTFSyw2RUK7bx6ciKayunAOq9hhUMBTJX4lyQpep7Pj6q6T8E6QDtmjMXR1x34uGaw4KwZxWOsgBS6PRAX3
HrzCQ7b2gLi5RY4IjWZvzldZXoKWKdupF9wHnZIpS8eLw20shFkYMh" "lPn0XhuooVNxpvKxLGxXpZAOxmwXUGpaqY4wD8w769mmHlSmGmlKRo1aDuSOPbqfOJTcPVks5hQFhpRsonoTKzUdOP6PKAZMHubU
ZHLlYRF9FqJDeRz7aCEGNe5utS9cOZIH4Unfh3iBUh23eOjMOHuFdxRpce1PQ42TNnUxueC9ied2OB5V8uhLAT9Ulelljc8GXxAK" "heF5TeWYKYlewgOMasKj60vcpma1lbvlydkpHe8uXiyIpza6nTgflJ
ow5osd9pwNnywZL7MRwdsue3LvKdH4M2FvKwL2UDRoBIH8lJUYGfgZLDYTDJHHIwaUB8RNOkpZwB1IiUAXU1CAtfkVKnwLB8Z15PxQIHHK2w554IZFqPDgzHW706b1tn0A74cS3IDuoOvTkYms" "TFWcgJ9B
OG4bFHsTnX3XwdYHUXNNPWTVeOM0GfXnATJJh49dqrC2BxCSXI21B26JiduUTIFOBF0ATSv0wPA07IMFAT6pBieK8nwRqaxvNY8kHs0uS1qAbJfiEBIqt8uAM5LS85ObIwZvFObwmF1jb9SIl6EYEcGXwBivB
tn20VSZviOPizauh2AmB0qlEFfpOCnOVvJu" "xS2kvafgg4DPc2IgG4zPTL1sTWsBrqp2eLLPLgddYqKXfbuGAYCjvXaLbyMGA3L19HYYN0EeBTDhCeiTBD5onoCQ2CsPvUkRKU1xktPg76tyUISDTSM9ZWS
Rx1jQ3nwxAWsXg88qYy2FIWjuhyX3te9cAaVBiG44NpDlmwl7UmqsQTRgaPDRwBOvDJSFf27ueVxaPn5F" "y4pZ5sWqoz1ewZvfdddQERHrUKLmRODlVRqZjcpWS1Ys0YIVGv8vt5W6f5oVZ69a03KC6kj1c
vhIiO4ahTeq4SZStVYx2GYbtcK38YL4WP5QXKwfh6Ww8z7mfYN0xwoBo3vc2FVgFNMEn9IcZqHNeEtKtnRaUGq2miPmNoloRRQq58ewhSraanU2qIGDwvQu7RStVFPF" "xvjWea2LdDobjFFS49OxrzRjgE8
IwVQtPOK9oQDCwBmuC7YAzxHsz1IzppV6yPEkofkDzph3lElpQZVZkoGs1x7atvjYuGZMNCVaXlLGLlhLB54QcRI40qGXVgYtqA3FIGDuRKfvy51RhyDzoJuXPevdxUVuEWlSctTjKsBZUoDdgmYXJrO5J62c
ITRLEYpFSPYOugMf" "UENAQGkHVmhD2b9wCS1K6WilzjtVTn9W1P5DI8VElOoO4f84Y4EzKjKrlMN6kXZLjkWN8x3xhHVsG78Su0AXlFuG0wIk0m6skHpkMe55LlH3Zb422qu4xvuxvwRuz1RZDNdypnlHDE
I5q9rVHmE0sNagyvcW5byoHA8AsiYYVRESo4XwqcE4XZrQv86FbLZWsXO8qUME" "lTwWpwXUybys6TISZDWnBPlPLp2i2hP9C6poA2FvBp5OWFwPMByntIU1nAQfZcdmGk1B7J04fO0lbl3kG4uZdmOlQaia
p4Leu30E7uHahsaWi7i8dD7Ro33GFr4OJ6Dx2mvV4xEgdGXMMnH4ufcel0w4mvuFkhXQiGo0Vpoe5Q8iq234avdPvsr9LJypDeMiqSCD7Xb2" "A6FQnCnoUtEcPStefm9pUGLmpyhN"

;; AUTHORITY SECTION:
pvabswoude.practicum.os3.nl. 3600 IN    NS      ns1.pvabswoude.practicum.os3.nl.

;; Query time: 45 msec
;; SERVER: 145.92.25.11#53(145.92.25.11)
;; WHEN: Sun Jan 28 15:31:40 2007
;; MSG SIZE  rcvd: 2048

patrick@patrick-desktop:/media/sdb5/rp1/code$
```

Figure 4.4: DNSSEC and UDP

4.6.4 Good and Bad things regarding DNSSEC

The goal of our research wasn't to "hack" the DNSSEC protocol, it was simply something we ran into during our research. We are convinced that the introduction of DNSSEC has solved one problem, but created another.

It solves the problems around DNS spoofing and DNS cache poisoning. But on the other hand, DNSSEC had enabled miscreants to create much bigger replies because the maximum UDP packet size is heavily increased. The question "Why?" is not hard to answer – they just needed more space for an answer. Still, it's a new protocol, why don't use TCP? We think they chose for UDP because there aren't many nameserver who talk TCP. Further more we think that there are many firewalls that have DNS with TCP disabled for security reasons. Another reason for using UDP, is because the protocol is much easier to implement into currently running nameserver. Further more, UDP is more lightweight than TCP.

We are afraid that the upcoming DNSSEC hype will cause for more problems around DNS DDoS attacks. This means that there should be more attention for security when implementing DNSSEC on CORNs.

4.7 An actual DNS DDoS attack

Alongside of all this theory we actually tested it as well. In the following table, you will see how we tested with various TXT records and noted what

Receiver packet size (B)	TXT record size (B)	+dnssec	protocol
497	420	no	UDP
590	497	no	TCP
604	513	no	TCP
1014	927	no	TCP
1025	921	yes	UDP
2033	1961	yes	UDP
2047	1959	yes	UDP
2048	1960	yes	UDP
2048	**1976**	**yes**	**UDP**
2065	1977	yes	TCP
2137	2045	yes	TCP

Table 4.7: Packet sizes and their protocol

protocol was used for the answer.

The first column of table 4.7 contains the size of the answer we got from out nameserver. The second column contains the "original" size of the TXT record. You might have noticed that there is a difference in bytes between column one and two. This is because an answer also contains an IP header of ∼50 bytes. The third column says if we enabled the DNSSEC flag or not. With the (*nix) command dig this can be done by adding "+dnssec" to the query. The fourth columns contains the protocol answer was wrapped in. If the first line of the dig answer is "`;; Truncated, retrying in TCP mode`" the answer is TCP, otherwise the answer is UDP. As you can see, we got a 2048 byte answer with the DNSSEC flag enabled.

4.7.1 DDoS bandwidth statistics

We also tested our theory by creating a victim and DNS DDoS that victim. We created a Perl script that was capable of spoofing an IP-address. Let's say you have a list of caching open recursive nameservers – which we had – and you are able to let those nameserver cache a (very big) record that you created – which we could do; with the DNSSEC flag enabled. And you got a victim. You could send a request to the caching name servers with the IP-address of your victim. The result is that the victim will get a lot of responses he didn't ask for from the CORNs which would take the victim out, or actually his Internet connection.

We executed 3 tests of 75 seconds (with the DNSSEC flag enabled) and where able to send out 100.000 requests! We had only one CORN and we installed a bandwidth monitor called bwm-ng[12] on the CORN so that we could monitor the incoming bytes per second (the spoofed request) and the outgoing bytes per second (the answer to the victim). To verify the results, we also registered the outgoing traffic on the spoofing server and the incoming traffic on the victim. We came up with the following results.

incoming: 148KB/s – outgoing: 5430 KB/s
incoming: 151KB/s – outgoing: 5670 KB/s
incoming: 149KB/s – outgoing: 5441 KB/s

We calculated that for each byte that comes in, the query, the victim will get a answer that is **36-38 times greater!** This means you need about 2.7 - 2.8% of the bandwidth of the target you're going to attack.

We have to note that we cheated a bit. During our tests we encountered a security feature of BIND9. After 1000 queries the named.log states

"no more recursive clients: quota reached"

This means that a BIND9 CORN only accepts 1000 requests from cache from 1 IP-address. And what do you know; 60% of the nameserver are BIND9[21] so this is a problem. This makes it hard to DDoS someone because you have to switch per 1000 requests to another nameserver. We researched if this can be influenced by external factors but have not found anything to influence this from the outside. On the other hand, if you search the Internet on the warning, you will find various forums[13][14] where administrators advise to set the query limit from cache higher (around 10.000) to avoid errors – talk about a workaround. On the other hand, concerning the total amount of CORNs we found, we don't think it's hard to create a script thats capable of switching nameserver every thousand requests.

Chapter 5

How to defend yourself

5.1 Introduction

In the previous chapters we described what reconnaissance work is most likely done by miscreants. We also described what problems we had during there reconnaissance work and we give an overview of some CORN related statistics.

In this chapter we will describe what can or cannot be done by administrators to rule out (recurring) abuse of their DNS services. This problem can be located on the DNS server itself but also on connected firewalls and/or routers. Because BIND9 is one of the nameserver that occur the most on the Internet [21], we will only describe security precautions for this kind of nameserver.

5.2 Securing your nameserver through BIND

The DNS Measurement Factory has conducted a DNS survey during august 2006 [21] and their conclusion was that BIND9 is running on 60 percent of all nameserver. BIND8 is on the second place with 13 percent. This means that the product we all know as "BIND" is running on 73% of all nameservers. Because BIND is the leading nameserver software on the Internet we will only give some advise how to improve your BIND configuration although the same ideas can be implemented in other DNS server configuration.

We got a lot of information from Rob Thomas' Secure BIND Template[22] and gathered information from other sites on the Internet and RFC's. For any configuration examples concerning BIND we refer to Rob Thomas' BIND Template[22].

5.2.1 Disable Open Recursion

In our research, we checked how much percent of the nameserver related to the .edu and the .int domain, are open recursive. We hope you've noticed, the results were shocking. In the DNS survey of august 2006 the DNS Measurement Factory estimated a number of 9.000.000 nameserver on the Internet. That means if ∼43% percent is really caching open recursive, that there are ∼3.500.000 nameserver who are just waiting to be abused.

The ironic thing is that Open Recursion in BIND is really simple to disable. Just add the following lines in you named.conf and Open Recursion is disabled for the masses of the Internet and enabled for the IP-addresses you define between the "{ }":

```
allow-recursion { address_match_list };
```

It's best to disable recursion for queries from the Internet and allow recursion from clients of your internal network.

5.2.2 Use ACL's

ACL stands for Access control list and enables a BIND administrator to create a lists of IP-addresses to determine what can or cannot be done. You could, for example, create a ACL of trusted IP-addresses from your internal network. This way you only have to edit the ACL if you want to change your settings thus have a more controlled way to configure your nameserver.

5.2.3 Create Views using ACL's

The best thing to in your BIND configuration is to create views with an ACL list. This way, an administrator can create a different 'view' for every occasion. In this way they could create a view for users in their network and create a different view for people who query their domain. This means that each view contains different properties so for example, your network clients get can use the server as recursing nameserver and the rest of the clients can't. A view can contain a reference to an ACL which makes it easier to control you configuration.

5.2.4 Define some 'standard' options

Of course there are also some standard options that can be defined in BIND. With standard options we refer to things as:

- Pid-file location.
- Dump file location.
- Notify yes/no for transfer requests to secondary nameserver.

- Allow transfers from ...

These configuration options might seem unimportant but are certainly part of a complete and safe BIND configuration.

5.2.5 Get your logging straight

With all above security precautions there is, of-course, still a chance that your are the victim of malicious activity. After all you are still involved with users who (unintentional) could abuse your nameserver. Therefore you must make sure you are able to backtrack what happened or what's happening in case of an attack. This is where logging comes in. Without the log, you could never check what happened thus you can never create security precautions against recurrence abuse.

5.2.6 Further security precautions

There are more security precautions to make sure your nameserver is secured. First of all, always make sure you're running the latest version of your nameserver software. Their not bringing an update because they think it's fun - it's there to close certain security breaches.

Also make sure youre nameserver software is running chrooted. This means you create a user 'bind' and a group 'bind' who only have rights in the 'named' directory. This way, when there is a security breach in nameserver the miscreants only have the same rights as the nameserver user has so the only thing they can do is shuffle your bind configuration - but of-course, like a good administrator, you will make backup's of your configuration.

Determine the right role for your nameserver. Because there are various roles for a nameservers2, you have to make sure that the nameserver has the role you defined. If, for example you only want a caching nameserver make sure your not running as an primary nameserver. This kind of wrong configuration could create various problems and above all a security risk - you simply didn't define the proper security precautions for a primary nameserver.

The most efficient way to setup your DNS configuration is to setup two nameservers with two views on each nameserver. The first view is focused on your internal network, has recursion enabled and is non-authoritative. The second view is focused on serving your zone to the outside world, is iterative, authoritative, has recursion disabled and doesn't cache. Depending on the size of your organization, you could of course create two nameserver focussed on your internal network and two nameserver focussed on serving your zone to the outside world but that's all up to you.

5.3 Securing your nameserver with a –not– nameserver solution

5.3.1 Securing your role with bots in your network

In section 4.5 we explained that a DNS DDoS attack works by sending a query to a CORNs with a fake source IP-address. When a attacker wants to execute such an attack, we determined that such a person need the possession of a bot-net to create sufficient bandwidth. As an administrator, you can't really be certain that you don't have infected computers in your network. To prevent that attackers could abuse your network computers for a DNS DDoS attack you could set some kind of source IP check in your firewall. This would mean for every packet that a client wants to send to the outside, the firewall will check if it has a valid (i.e. is the source from my local network) address before the packet is actually transmitted to the Internet. This way you can make sure that your computers aren't being used in an attack because the fake source IP-address will not exit your network.

5.3.2 Securing your role when you intentional run a CORN

Some administrator intentionally run a Caching Open Recursive Nameserver. This could be done for various reasons like:

- Make sure that your external users always have a 'safe' nameserver.
- Legacy reasons – It could be that a nameserver configuration has been an ORN in 1980 (when the Internet was safe) and with updates the configuration has stayed the same.
- They just want users to use there nameserver to make themselves feel good.

The problem is, when running a CORN, you could be the victim of abuse. It can be safe to run a CORN but you have to meet a few security precautions. First of all, like in section 5.3.1, your firewall has to check the source address of the packets towards the nameserver. When a source IP comes by for lets say 400 times a second for 10 seconds long, you could let your firewall drop those packets because it isn't very likely that one client queries a nameserver 4000 times within 10 seconds – we assume that no human is capable of doing that.

5.3.3 Securing your role as a potential victim

The only advise is: don't piss people off so they have a "reason" to attack. If you are a potential victim there is not much you can do you just have to hope that other people have their configuration straight. The problem with a DNS DDoS attack is that your Internet connection is filled with UDP packets. You may have some sort of firewall, it doesn't help because your line will still be flooded with UDP packets. You could make sure there is a backup connection but you would have to change your DNS settings to be available again for you users – but remember that miscreants could also find you again this way.

5.4 And if you really want to run a CORN...

In section 5.3.2 are some reasons of running a CORN on the Internet. Lets say you want to run a CORN because your the administrator of an international organization and you want your users to have a secure nameserver to use when they are out of office. The safest thing is to let you nameserver allow recursion from you internal network and disable recursion from the Internet. If your users want to use your nameserver, let them connect through a Virtual Private Network to your network. This way your users will actually be **inside** your network so they can use your nameserver.

Chapter 6

Conclusions and Future Work

6.1 Summarized results

As read in this report, we conducted research to Caching Open Recursive Nameservers during the month January of 2007. In this period we researched how many nameserver of the .edu and .int zone are CORNs and what the impact could be on the Internet society when CORNs are abused. We gathered the following statistics concerning the .edu and .int zone.

6.1.1 Reconnaissance work

In table 6.1 a summarized survey gives us a good impression of the total CORNs.

Zonefile	ORNs	CORNs	CORNs (no time-outs)
.int (inside .int domain)	36%	36%	41%
.int (outside .int domain)	33%	32%	33%
.edu (inside .int domain)	51%	50%	64%
.edu (outside .edu domain)	45%	42%	47%

Table 6.1: Global statistics .edu en .int zonefile

In table 4.6 the totals of all test are summarized (the abbreviation NS refers to nameservers). An average of 41% of the nameservers in the .edu and .int domain is a CORN. If time outs are excluded it's an average of 48%

6.1.2 DDoS bandwidth statistics

We executed 3 tests of 75 seconds (with the DNSSEC flag enabled) and where able to send out 100.000 requests. We came up with the following results.

incoming: 148KB/s – outgoing: 5430 KB/s
incoming: 151KB/s – outgoing: 5670 KB/s
incoming: 149KB/s – outgoing: 5441 KB/s

We calculated that for each byte that comes in (the query) you the victim will get a answer that is **36-38 times greater!** This means you need about 2.7 - 2.8% of the bandwidth of the target you're going to attack.

6.2 Do we have to be concerned of large DNS DDoS attacks using CORNS?

In short – YES! We had some surprising yet shocking results during our research period. We have experienced that it's not hard to create a list of nameservers. This is because nameservers provide a common service which is needed all around the world, by almost every piece of software created. But hold on. It's not the nameserver we have to worry about, we also noticed that a lot of nameservers (∼41%) is a CORN – and that's where the worries should come in. The DNS survey of The measurement factory[21] estimate around 9.000.000 nameserver on the Internet. Based on our research, we estimate that there are ∼3.500.000 CORNs that could be abused. Abusing a CORN isn't very hard and because there a lot of them out there and the protocols that are being used make it easier.

6.3 Future Work

We have encountered that DNSSEC also helps us to increase the UDP packet size[1]. What we didn't do in our research is locate how many nameserver have got the DNSSEC extension enabled[2]. Some future work around this subject could be useful.

In 4.4 we've explained that we we haven't done sufficient research concerning forwarders. We have worked a way to 'test' if a nameserver is a forwarder, and we've created a simple test to see if the answering NS is also a CORN. Some future work concerning forwarders could be a useful addition to our CORNs research.

Is there a way to conduct the same kind of attack with an other kind of RR than a TXT record? It could be for example that the "Additional Section" of an answer can be influenced or is bigger with some domains/queries.

[1]See section 4.6.3

[2]We installed a standard BIND 9.3.2 server and the DNSSEC extension was enabled by default.

Some future work around –non– TXT records could be useful as extra information around DNS DDoS attacks.

Is there any way that you could use *ORNs* in a DNS DDoS attack. Normally, you would want to stay hidden because you will let the CORNs cache a record from your hacked nameserver and then throw your hacked nameserver offline, so you can't be traced. There could be done some research if there any situation that you still could use the ORNs in your DNS DDoS attack, but still remain undetected.

Bibliography

[1] P. Mockapetris, *Domain Names - Concepts and Facilities*, Nov 1983, `http://www.ietf.org/rfc/rfc882.txt`

[2] P. Mockapetris, *Domain Names - Implementation and Specification*, Nov 1983, `http://www.ietf.org/rfc/rfc883.txt`

[3] P. Mockapetris, *Domain Names - Concepts and Facilities*, Nov 1987, `http://www.ietf.org/rfc/rfc1034.txt`

[4] P. Mockapetris, *Domain Names - Implementation and Specification*, Nov 1987, `http://www.ietf.org/rfc/rfc1035.txt`

[5] Microsoft Technet, *Windows 2000 DNS*, `http://www.microsoft.com/technet/prodtechnol/windows2000serv/plan/w2kdns2.mspx`

[6] PowerDNS, *A modern, advanced and high performance nameserver*, `http://www.powerdns.com/`

[7] MaraDNS, *A security-aware DNS server*, `http://www.maradns.org/`

[8] Wikipedia, *Return on Investment*, 21 Jan 2007, `http://en.wikipedia.org/wiki/Return_on_investment`

[9] Rich Miller, *Widespread Outages for World of Warcraft*, 25 Mar 2006, `http://news.netcraft.com/archives/2006/03/25/widespread_outages_for_world_of_warcraft.html`

[10] Nate Anderson, *Vint Cerf: one quarter of all computers part of a botnet*, 25 Jan 2007, `http://arstechnica.com/news.ars/post/20070125-8707.html`

[11] R. Elz & R. Bush, *Serial Number Arithmetic*, Aug 1996, `http://www.ietf.org/rfc/rfc1982.txt`

[12] Volker Gropp, *bwm-ng (Bandwidth Monitor NG)*, 20 Feb 2005, `http://www.gropp.org/?id=projects&sub=bwm-ng`

[13] brandonhutchinson.com, *"no more recursive clients: quota reached"*, 16 Feb 2006, `http://www.brandonhutchinson.com/no_more_recursive_clients:_quota_reached.html`

[14] webhostingtalk.com, *"no more recursive clients: quota reached"*, 12 Aug 2004, `http://www.webhostingtalk.com/showthread.php?threadid=351379`

[15] A. Newton, *A Lightweight UDP Transfer Protocol for the the Internet Registry Information Service*, 9 Jan 2007, `http://tools.ietf.org/html/draft-ietf-crisp-iris-lwz-07`

[16] R. Rosenbaum, *Using the Domain Name System To Store Arbitrary String Attributes*, May 1993, `http://www.ietf.org/rfc/rfc1464.txt`

[17] Wikipedia, *DNS cache poisoning*, 25 Jan 2007, `http://en.wikipedia.org/wiki/DNS_cache_poisoning`

[18] Spacefox (Secure Sphere Crew), *DNS Spoofing techniques*, 23 Jan 2002,`http://www.securesphere.net/download/papers/dnsspoof.htm`

[19] NLnet Labs, *A short history of DNSSEC*, 8 Jan 2007, `http://www.nlnetlabs.nl/dnssec/history.html`

[20] Wikipedia, *Transmission Control Protocol - Connection establishment*, 30 Jan 2007, `http://en.wikipedia.org/wiki/Transmission_Control_Protocol#Connection_establishment`

[21] Infoblox, *DNS Survey*, Aug 2006, `http://dns.measurement-factory.com/surveys/200608.html`

[22] Rob Thomas, *Secure BIND Template Version 5.4 12*, Oct 2006, `http://www.cymru.com/Documents/secure-bind-template.html`