Security Monitoring of DNS traffic

Bojan Zdrnja
CompSci 780, University of Auckland, May 2006.
b.zdrnja@auckland.ac.nz

Abstract

The Domain Name System (DNS) is a critical part of the Internet. This paper analyzes methods for passive DNS replication and describes the replication setup at the University of Auckland. Analysis of the replicated DNS traffic showed great dependency of collaborative anti-spam tools on the DNS. These tools also put a great burden on the DNS. This paper discusses analyzed anomalies in the replicated DNS traffic: typo squatter and fast flux domains, private IP address space leaks and non recommended characters in DNS names. Future applications of passive DNS replication are also discussed.

1. Introduction

The Domain Name System (DNS) is a critical part of the Internet. Most Internet applications used today need the DNS to function properly. Although the Internet can function without the DNS, as only IP addresses are needed for establishing communication links, one can not expect users to remember IP addresses. The DNS, which translates domain names into IP addresses (and vice versa), is therefore critical.

In order to provide scalability and availability, the DNS consists of multiple databases located on servers which are authoritative for their zones. Besides legitimate applications, various malicious programs and other security attacks depend on, and often abuse the DNS. As no single machine on the Internet has a truly global view of the DNS, it is difficult to analyze the DNS traffic in order to detect anomalies or potential abuses.

This paper describes methods for passive replication of the DNS data, not only DNS replies but also DNS queries. After a brief overview of the DNS is given, the paper presents architecture deployed at the University of Auckland and observed anomalies of the captured DNS traffic.

2. Background

Since the DNS has to support a potentially unlimited number of domains (which can be mapped to a limited number of IP addresses – 4 billion in IP version 4, assuming 100% assignment efficiency), its architecture has to be extremely scalable.

In the beginning [1], the whole DNS database was stored in a single file which was located on every computer on the Internet. As this single file had to list all the computers on the Internet (so the local machine could translate a computer name into the corresponding IP address) it became impossible to manage it as the Internet grew.

A logical move from a local static file is to implement a database. However, as the DNS is critical for normal operation of the Internet, there should be no single point of failure.

The DNS therefore consists of various databases. As the DNS name space "is a variable-depth tree" [1], each organization is given a specific zone in the DNS hierarchy. At the top of the tree is the root domain (".”). The "." character is also used to mark the boundary between different zones.

Mappings that are stored in DNS zones are called resource records (RR). Resource records are further distinguished by their type [2]. The most common resource record types used include A, which map domain names into their IP address, CNAME which identifies canonical names (aliases) for domain names, MX which identifies mail exchangers for the domain (servers that will accept e-mails for queried domains), NS which identifies authoritative name servers and PTR which identifies a pointer to a domain space.

Each administrator has to provide the data for the zone they are authoritative for [2]. As each zone is a separate database, a misconfiguration of any one zone does not affect functionality of rest of the DNS. However, this also means that the data contained in a zone file is visible only to the local administrator.

It is possible to transfer a whole zone from a DNS server. Zone transfers are normally used between the master (primary) server and other secondary (slave) servers for this zone. It is a security recommendation that zone transfers are limited only to legitimate servers for a domain as this information can be valuable to potential attackers. Most of the DNS servers today will not allow zone transfers from remote machines.

An external user could also, theoretically, brute force all potential domain names in order to see which exist in a particular zone, but this operation would not only be intrusive, but also resource intensive. This method would also be valid only for forward resource records (records which translate domain names into IP addresses).

There exists a special domain IN-ADDR.ARPA which should be used for "gateway location and Internet address to host mapping" [3]. There are various problems with reverse resolution. Administrators who provide the information for forward queries, and who effectively own their particular zone, rarely own their IP space in the IN-ADDR.ARPA domain; these are typically owned by Internet Service Providers. This means that the records in the IN-ADDR.ARPA domain are quite often outdated or even non existent. It is also possible to have multiple forward records mapped to one IP address. As the records in the IN-ADDR.ARPA domain can only have one mapping, it is impossible to find all the forward mappings to this IP address with any active DNS query tools – the only way would be to see the zone data file.

2.1 Multiple mappings

Multiple mappings to a single IP address are very common today. These are most often used on web servers. With the introduction of the HTTP 1.1 protocol, which is supported by all major browsers today, a Host header field was added as a requirement for all request messages [4]. This header field makes it possible to host multiple Web sites on one physical machine, with only one IP address. The main reason that the World Wide Web Consortium introduced this header field was conservation of the IP address space.

It is typical today that web hosting providers host multiple web sites on one physical machine. The hosted web sites, which can potentially have any domain name, just have to point their forward mappings to the IP address of the server in the hosting company. As any domain name in the DNS can point to any IP address it becomes impossible to enumerate all the web sites which are hosted on one IP address.

2.2 Domain name changes

Each DNS reply has a Time-To-Live (TTL) field. The TTL field tells the client resolver how long it should cache the answer. While the value is cached by the client, it will not ask any DNS server when some application needs this information again. Once the cache

expires, and if some application issues the same query, the client resolver will send standard DNS queries to obtain new information.

Historical values of various domain names are not stored anywhere. The local resolver will remove this information as soon as the time-to-live has expired. When the domain name is changed by the authoritative server, it is no longer possible to view previous data, unless the local resolver was reconfigured to log all queries and their respective answers.

This is especially important with fast flux domains. Fast flux domains [5] are domains that change their resource records many times in short periods of times (sometimes minutes or hours); they will also have low TTLs. Low TTLs will automatically force client resolvers to resolve the fast flux domain name often instead of using the cached value. These resource records are then used to control malicious programs which are actively infecting computer systems on the Internet. An attacker usually sets up their Command and Control (C&C) server to which all the infected machines (also called "zombies") report; it is then easy to control those machines and instruct them to do whatever the attacker wants.

Early versions of malicious programs had hard coded IP addresses which made them easy to stop; an ISP could just drop all traffic to the IP address of the C&C server to stop further control of infected machines.

Malicious authors today use fast flux DNS domain names. They register a special domain name which is then pointed to their current C&C server. In case that the server is disabled, the attacker just has to change the DNS resource records for their domain to a new C&C server. As the TTL of resource records was small, it will not take long until infected machines send new DNS queries and connect to the current C&C server.

The only way to stop C&C servers which are using domain names is to disable their DNS domain, which can be difficult depending on the registrar that the attacker used in order to register their domain.

In cases like this it is interesting to see the full history of a fast flux domain because most attackers will use compromised systems as C&C servers, so a list of servers that the fast flux domain pointed to can be used to alert the system owners.

Spammers also use fast flux domains to deliver their spam e-mail messages. The fast flux domains get registered for a very short period of time, so the spammers can deliver their

messages properly. These new domains are used in the “From:” fields of messages, so they will pass the domain tests most SMTP servers perform today. Domains which are under spammers’ control can also be configured to pass tests performed by the Sender Policy Framework [6] system (SPF). Once the spammers have sent all the e-mail with a particular domain in the “From:” field, the domain is typically unregistered. An historical view of the domain pointers and queries allows easier analysis of network traffic (in case of bot machines on the local campus) and inconsistencies in the SPF records.

2.3 DNS cache poisoning

Poisoning of the DNS cache is a more advanced technique that attackers use to redirect victim’s traffic to a different server. This is possible as a result of the transaction ID of a DNS query. According to the RFC 1035, the transaction ID is “a 16 bit identifier assigned by the program that generates any kind of query.” The transaction ID must be returned in the reply, so the client resolver which generated the request can match replies with queries it sent. When a client sends a request to a remote server, it will initially send a UDP packet with source IP address of the client, a randomly generated source port and the destination IP address of the DNS server; the destination port is always 53. In order to identify the DNS request and reply, the client will set the transaction ID, which is the “sole form of authentication for a DNS reply” [7]. If the client’s DNS server supports recursive querying, it will do the rest of the querying on its behalf.

An attacker has only to guess the source port number and the transaction ID number (as he already knows victims DNS server, and the IP address of the authoritative DNS server he wants to spoof) in order to create a spoofed packet which will be accepted by the victim’s DNS server. This attack is called DNS cache poisoning.

If the attacker can use a possible victim’s DNS server to perform his own DNS lookups, as is the case with misconfigured DNS servers which allow recursive DNS queries to be performed by any machine in the Internet, it is even easier to perform DNS cache poisoning. In this case the attacker does not have to wait for the victim’s machine to send the DNS request.

Although today’s DNS servers perform randomization of the transaction ID, it is possible to predict the randomization sequence. DNS servers are susceptible to two types of attack: birthday attacks and phase space analysis spoofing [7].

In birthday attacks, an attacker has to send a sufficient number of spoofed packets to the victim's DNS server in order to poison the DNS cache. If the DNS queries on a network are monitored, these attacks can be spotted easily as they will consist of multiple reply packets for a single domain with various transaction IDs sent in a short period of time.

The second attack, phase space analysis spoofing is based on Michal Zalewski's work on the predictability of TCP sequence numbers [8]. The same analysis can be applied to DNS transaction IDs. Although transaction IDs should be random, in some implementations it is possible to predict these numbers, and attackers can use this vulnerability in order to poison the DNS cache.

3. Related work

At the FIRST 2005 conference, Florian Weimer presented his passive DNS replication project [9] that captured DNS traffic and stored it into a local database for later analysis. Weimer's project, *dnslogger*, consisted of sensors deployed across the network and analyzers at a centralized database server. Sensors captured real time DNS traffic and forwarded it to the analyzers which stored the data into a database. Weimer gave examples and reasons for using passive DNS replication techniques. The main result of his analysis was the detection of botnets and some active Denial of Service attacks performed by abuses of DNS resource records.

Weimer noted that the replicated data is extremely localized. Although this sounds surprising, it can be expected as only the DNS traffic of a particular group of users is observed. In Weimer's case, the deployment was at the University of Stuttgart. Users at campuses like this will observe a certain pattern of interest, which will lead to the replicated data that is localized either to their interest or geographical location.

Another project based on Florian Weimer's work is Schonewille at al's research project at the University of Amsterdam [10]. The goal of their project was to monitor and detect actions of malicious programs in their network. The main capturing engine for their project was same as the one in [9], however, as their goal was to identify local machines which are infected, they were replicating outgoing DNS queries and not replies. Using data analysis methods described in the paper, Shonewille was able to detect some infected machines. The data logged in this project has to be analyzed carefully as there

are privacy concerns due to the logging of source (requesting) IP addresses. It is worth mentioning that by correlating data logged in both projects; it should be possible to detect C&C centers as they typically behave like fast flux domains.

John Kristoff wrote the DNSwatch software [11], and Elton at al of the College of William & Mary mentioned the possibility of using this software to detect infected machines on a local network in their paper "A Discussion of Bot Networks" [12]. Their work is based on a known black list of servers which are used to spread malicious programs or to run C&C servers. The DNSwatch software is then used to parse the DNS logs in order to detect infected machines. As DNSwatch parses logs of the local DNS servers, this approach works only if infected machines use their local DNS servers in order to resolve DNS names. A method similar to Schonewille's, which monitors DNS traffic on the Internet link is needed in order to detect clients resolving names using other DNS servers than those officially provided.

Ishibashi et al [13] used DNS traffic to detect mass mailing worm infected machines. Their approach is based on the number of queries issued for MX records which indicates machines that are trying to send e-mails to different domains.

4. Passive DNS replication implementation

In order to set up a monitoring environment, data sources for DNS traffic have to be identified. As noted in Weimer's paper [9], there are several data sources that can be used in order to collect DNS requests and replies. Depending on the nature of the data that is monitored (DNS requests or replies); there are several possibilities that can be implemented. The deployed environment will depend on the data source that was chosen. The list from Weimer's paper was expanded; advantages and disadvantages of each of the data sources listed are discussed below:

- *Periodical polling of DNS servers*. This method is the most intrusive method. The domains that are polled have to be known in advance; otherwise a brute force method has to be used in order to enumerate resource records in a DNS zone. It is sufficient to say that this method is therefore very impractical – it requires a lot of network resources as each DNS query has to be sent to an authoritative DNS server, causing a significant impact.

- *Perform zone transfers*. Zone transfers offer full information about a particular DNS zone as they contain all resource records. As Weimer noted, this requires at least some degree of cooperation with the target DNS system administrators. Most of the DNS software today by default does not permit DNS zone transfers due to possible abuses of this information. Such servers refuse any zone transfer requests. It is indeed possible to find DNS servers that are either misconfigured or deliberately open to zone transfers, but this should not be relied on.

  It is worth mentioning that some of the top level DNS zone providers offer free download of zones they are hosting. This does not include standard DNS zone transfers (AXFR), as defined in RFC 1034, and later expanded in RFC 1995, which defined incremental zone transfers. These mechanisms usually rely on file transfers using external software, such as FTP or rsync. VeriSign, one of the registrars for the .com and .net domains, offers a "TLD Zone Access Program" [14]. The authorization to access this program has to be requested from VeriSign and, once granted, a user is able to use FTP to download the zone files for .com and .net top level domains. These files contain only active domain names for each of the top level domains and are updated twice per day. As the number of active .com and .net top level domains is in the tens of millions, these files are very big (over 100 MB). VeriSign does not support any incremental downloads so these files have to be downloaded in full every time, which makes this approach impractical.
- *Modify client DNS resolvers*. Each client machine has a local DNS resolver library which is used when an application needs to query a DNS server. This local DNS resolver first consults the local hosts file and it's own, local, DNS cache. If an answer is found, the local resolver will return the information to the application that requested it, and no other DNS queries will be performed. This means that it is entirely possible for a client machine to never send any DNS traffic if all DNS queries are already located in the local hosts file or in the local DNS resolver's cache file. In all other cases, the local DNS resolver will send the request to the preconfigured DNS server for this machine. By modifying the local DNS resolver library it is possible to log all DNS requests and replies that the client machine

issues. These logs would have to be forwarded to a centralized server in order to be processed and analyzed. There are two main problems with this approach.

First, modification of the client DNS resolver library is impossible or very difficult on proprietary operating systems, such as Microsoft Windows. As Microsoft Windows clients are usually the majority of deployed clients, this makes collection of logged DNS requests and replies on the client machines practically impossible.

Second, there are privacy issues with this method of collecting the data. Although the privacy issues are present in other methods as well, modification of the local DNS resolver makes it very easy to identify all DNS requests and replies originating from/to one client machine.

- *Modify server DNS resolvers*. This method relies on modification of the local DNS code so the DNS requests and replies are logged. As most of the DNS servers deployed today are BIND DNS servers [15], which is an open source DNS server application, it is relatively easy to modify the program source so all the DNS requests and replies are logged.

  One of the disadvantages of this setup is that only requests coming to the local DNS server will be logged – clients that are configured to use any other external DNS server will be missed. Most corporate environments block outgoing or incoming DNS traffic which is not destined to the official DNS servers, so this method is valid for such environments.

  The privacy problem is present here as well. If the DNS requests to a particular DNS server are logged, without source IP address obfuscation it is possible to trace certain machine's DNS requests.

- *Passive DNS replication by capturing network traffic*. By capturing the DNS packets on the network, it is possible to overcome problems associated with the previous methods. In this case, clients which are configured to use an external DNS server will have their DNS requests and replies logged properly, if the DNS sensor is capturing the Internet traffic.

  By capturing the DNS traffic at the Internet link point, most of the privacy problems are already solved. As the client machines should be using internal DNS

servers as their resolvers, and these servers should be used recursively (meaning that the internal DNS servers should try to completely resolve the query and send back to the client the final answer), the DNS traffic at the Internet link point will be between the internal DNS servers and other DNS servers on the Internet. Exceptions for this are internal clients which are directly using external DNS servers. In this case the source IP address of DNS queries and the destination IP address of corresponding DNS replies will reveal internal machine's IP addresses.

Depending on the expected outcome from the DNS traffic monitoring, internal IP addresses can be anonymized. The easiest way to anonymize this traffic is to use Minshall's *tcpdpriv* utility [16], which is used to eliminate confidential information from collected network packets.

4.1 DNS traffic parsing complexities

Capturing the DNS traffic properly is a complex task due to various enhancements and add-ons on the original protocol described in [3]. DNS traffic generally uses communication on UDP port 53. RFC 1035 also restricts messages in UDP traffic to 512 bytes. If the response is longer than 512 bytes, the replying DNS server should set the TC (truncated) bit in the DNS header. Section 4.2.2 of RFC 1035 describes further how to carry packets over TCP with the limitation of 65535 bytes on the maximum message size. An RFC document released two years later, RFC 1123 [16], "Requirements for Internet Hosts – Application and Support", expanded the use of DNS traffic in case of replies longer than 512 bytes. This RFC defines the use of transport protocols in DNS and states that TCP must be used when performing DNS zone transfers (due to the large size of data being transferred). It also recommends that TCP is used as a backup communication mechanism when messages exceed 512 bytes in size (and the truncation bit in the reply is set). In this case, the resolver should switch to TCP and reissue the query; the TCP communication will enable it to get the full answer back.

During previous work on the DNS protocol, the author of this paper has noticed that most of the resolver libraries today (at least those implemented in Microsoft Windows and Linux operating systems) will indeed fall back to TCP when they receive a UDP reply with the truncation bit set. However, in some cases, when the resolver library decides that it received enough information about the requested query (for example, if the client

application queried the A resource record of a certain machine name), it may not send additional requests over TCP.

To make things even more complex, a recent RFC [18], "Extension Mechanisms for DNS (EDNS0)" defined a new opcode field pseudo resource record which allows DNS traffic to be carried in UDP packets longer than 512 bytes. This resource record is called pseudo because it just specifies that the UDP packet carries more data than 512 bytes (the RFC does not define any new DNS data). Previous analysis of the DNS traffic again showed that a lot of today's implementations of the DNS server software, such as BIND, already support EDNS0.

All these modifications to the original protocol and various implementations make DNS parsing difficult. When analyzing network traffic, it is relatively simple to identify DNS traffic, as the source port will always be 53, no matter if the communication is over TCP or UDP. It is worth noting that some implementations of the DNS server software will also use source port 53 when sending queries; in this case both source and destination ports in a communication will be 53. Once the traffic is identified, it has to be properly parsed and, depending on the implementation used; this can be more or less difficult.

4.2 Implementation architecture

After researching possible methods for monitoring the DNS traffic in university or corporate environments, it was concluded that the best and most practical way is by capturing DNS packets on the network.

This method is applicable to most environments today. In order to support environments which have multiple Internet connection links, or high speed networks where network traffic processing at a single point might present a problem for today's equipment (for example, on 10 Gbit network links), a distributed architecture, similar to those described in Weimer's paper was picked. This architecture is described below:

- A sensor is a machine that is collecting DNS traffic from the network. The sensor forwards collected DNS packets to the logging server (the collector); depending on the implementation, the forwarding will be in real time (as soon as a packet is captured), or as a batch process, in regular time periods. Forwarding in batch processes is recommended for very busy systems as the data can be greatly

compressed, so the overhead of another transmission is reduced as much as possible.

- A collector is a machine which is collecting captured DNS traffic and processing it. Once the traffic is processed, it is stored in a database so it can be analyzed later. Depending on the environment and resource requirements, the database can be located on the collector machine, or there can be multiple collector machines storing the data in a centralized database.

4.3 University of Auckland passive DNS replication implementation

DNS monitoring implementation at the University of Auckland was similar to the one described in Weimer's paper [9]. A sensor was deployed at the perimeter, located in the DMZ. A different machine, the collector, was deployed in the local network.

Although the architecture was similar to Weimer's, the setup deployed at the University of Auckland was not real time: the DNS traffic was logged at the sensor and then transferred in batch to the collector. Initial implementation transferred logged traffic every hour as the master script rotated captured traffic files every hour.

Figure 1 shows the setup at the University of Auckland:

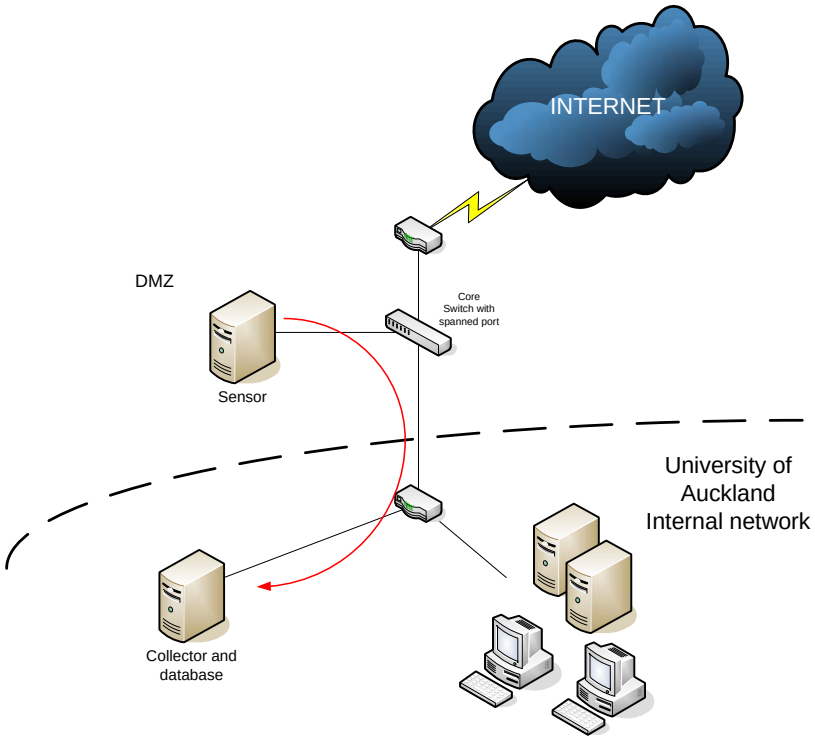

Figure 1: *DNS replication implementation at the University of Auckland*

As the University of Auckland has two Internet links which are both connected to the same router, one sensor machine was enough to capture traffic. The sensor machine was connected to a SPAN port on our gateway router. The SPAN port mirrors all traffic to/from both our ISPs to the sensor, which was configured to log only DNS traffic. Once DNS traffic was captured, it was transferred to the collector machine (sitting in the internal network) with a batch process.

The software used for capturing the DNS traffic was tcpdump [19]. Tcpdump is a standard network traffic logging program with a powerful analysis engine and filters. Filters enable a user to specify exactly what should be logged by tcpdump and were used in this setup so only authoritative DNS reply packets were logged. All other DNS traffic was dropped as it was not relevant for this project.

RFC 1035, in chapter 4.1.1, defines the DNS packet header. This header is shown in Figure 2:

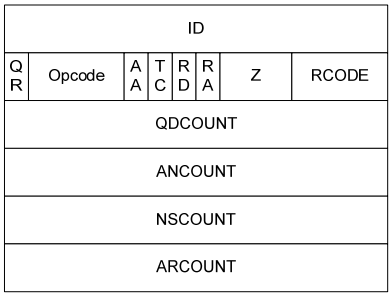

Figure 2: *DNS packet header (after [3])*

This header is present in every DNS packet sent or received. For passive DNS replication, of particular interest are DNS replies which have the Authoritative Answer (AA) bit set. According to the RFC 1035, "this bit is valid in responses, and specifies that the responding name server is an authority for the domain name in [the] question section."

For simplicity other fields of the DNS header are ignored and only replies with the AA bit set are logged. This was accomplished with the following tcpdump filter:

```
udp port 53 and ( udp[10] & 0x04 != 0 )
```

Figure 3: *tcpdump filter to log only DNS replies with the authoritative answer bit set*

Tcpdump logs traffic in libpcap format files which can be parsed with any other tool that supports this open source format for network packet captures. Logged files are periodically transferred to the collector machine. As tcpdump already has a powerful

DNS traffic analysis engine, it was used on the collector machine in order to parse the captured files and populate the local database.

This setup can be applied in any environment and is very scalable. As the files are processed in batches, the risk of any of the machines dropping traffic and loosing results is much lower. In Weimer's implementation the DNS traffic is forwarded to the collector machine as soon as it is captured – in case of a big network with more sensors pointing to one collector or analyzer machine it is possible to loose network traffic due to either machine or the underlying protocol, as the logged DNS packets are encapsulated in UDP traffic.

The implementation at the University of Auckland uses rsync over SSH in order to copy the logged traffic to the collector machine. This way, the transmission is both reliable and can be authenticated by strong cryptographic means.

The results are kept in two local MySQL tables. One table stores all successful DNS replies which have been parsed, with their RR type, time when the record was first seen and time when it was last seen. The other table stores all replies which had a response code of NXdomain (non existent domain). This allows us to determine the number of unsuccessful queries.

5. Experimental results

The experimental setup described in section 4.3 was left running for 2 weeks on the University of Auckland network. Although the information in the database is completely anonymized (it is not possible to identify which client machines issued which DNS requests), it is possible to identify types of network traffic that generate most of the DNS queries.

There were various anomalies detected, of which some of the causes were explained in section 2. Some interesting observations are highlighted in the following sections.

5.1 Query/reply rates

The University of Auckland operates two main DNS servers. These servers are authoritative for the auckland.ac.nz domain so all queries for machines within this zone will be directed to them. There are other DNS servers around the University which are authoritative for their own zones, and which are externally reachable.

Most of the clients inside the University are configured to use these two main DNS servers as their resolvers.

Logged DNS traffic showed very high rates of authoritative replies coming to the University of Auckland network. Although the typical pattern of the working week is still visible, the number of replies rarely goes below 70 queries per second and it's peeking at about 90 queries per second. **Figure 4** shows received DNS replies between Monday, 8.5.2006. and Saturday, 19.5.2006.

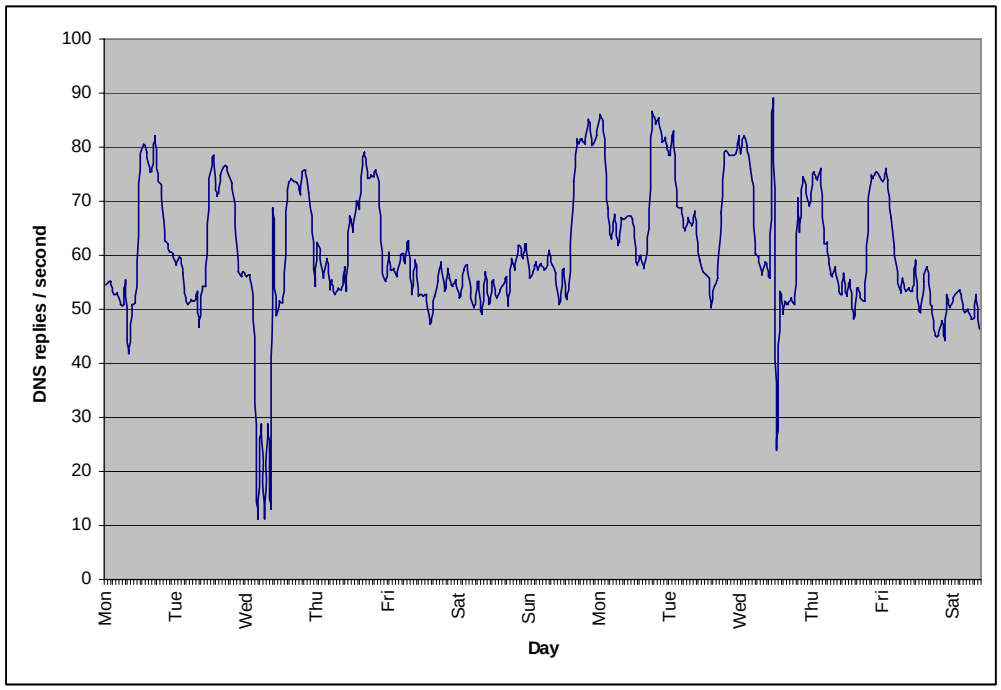

Figure 4: *Received DNS replies*

The main reason for a constantly high number of DNS queries is the mail processing gateways at the University. During the process of spam detection, these gateways send a large number of DNS requests in order to obtain additional information about the IP addresses e-mail is coming from. The mail processing servers use the SpamAssassin software [20] to detect spam, which contacts various RBL (Realtime Blackhole List) servers [21]. RBLs rely on DNS A records in order to identify machines which are sending spam e-mail. Machines that are known spam sources will have A records in zones which are dynamically modified on RBL servers. Because SpamAssassin checks all IP addresses which are detected in the e-mail being processed, including IP addresses found in the e-mail body, it generates a huge number of DNS queries. In fact, 28.9%

(1215580 replies) of all DNS A records logged in the database belong to various RBL systems. This was determined by querying the database for all domain names of the RBL DNS servers that are used with the current version of SpamAssassin [20] installed on the University of Auckland e-mail processing gateway. A similar number of 27.6% (2388813) records, that can be correlated with requests going to the RBL DNS servers is logged in the NXdomain database.

The graph also shows when the University Internet (world) link was down due to a scheduled maintenance. The other link, for New Zealand traffic only, was still up during the outage; DNS replies coming through this link are visible on the graph. The number of replies on the New Zealand traffic link is about 20 replies per second.

5.2 Query types

Query types for address records dominate monitored DNS replies. This is again partially caused by the mail processing gateway because all RBL DNS requests have the resource record type of A. Table 1 shows resource record type percentages for both databases:

RR type	**% total replies**
A	62.6% (2626759)
PTR	16.4% (690316)
MX	8.3% (347876)
NS	4.9% (205267)
CNAME	4.2% (159408)
TXT	3.6% (75076)

DNS replies answered with a RR

RR type	**% total replies**
A	64.2% (5546056)
TXT	31.4% (2710255)
PTR	4.1% (353811)
AAAA	0.5% (21417)
NS	0.2% (9429)
MX	0.1% (1437)

DNS replies answered with NXDomain

Table 1: *Percentage of RR types in DNS replies*

The high percentage of A RR type replies also matches behavior observed in [22], as most of the user activities are related to the WWW sites, most client machines will just try to resolve machine names and get their A RRs.

The unusually high number of TXT RR replies in the NXdomain database can also be attributed to the mail processing gateway. Sender Policy Framework [6] relies on the TXT records in order to identify e-mail servers which are legitimate for sending e-mails with particular domains in the From: fields. As the University of Auckland e-mail system is configured to use SPF in order to check forged e-mails, for each incoming e-mail these servers will issue a TXT RR query to the sender domain's DNS server. As SPF is still not widely deployed, this results in a high number of NXdomain replies.

5.3 Data locality

When the monitoring architecture was deployed, it was expected that the captured DNS data would be more localized, as was noticed in Weimer's paper [9]. This was expected because the user base is more or less constant, so after a period of time most of the interesting domains which are visited should be in the DNS database. Figure 5 shows statistics for the two week period that the monitor was running on the University of Auckland network:

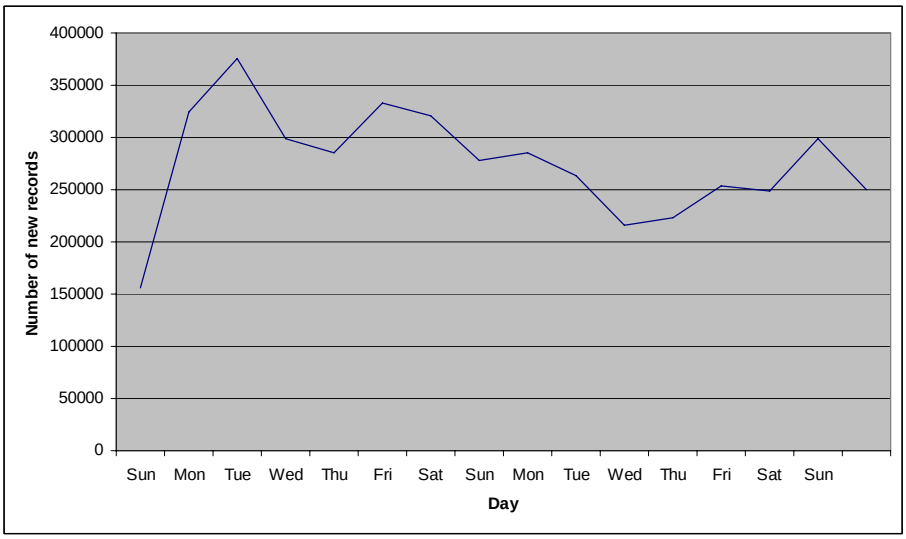

Figure 5: *Number of previously unseen records in the database per day*

The average number of new records in the database for the period DNS traffic was monitored is around 25,000. Although a particular site does observe a localized part of the whole DNS system, due to the actions of its user base, spam that is received, and which subsequently triggers multiple spam detection mechanisms as described in previous sections, causes the number of new domains per day to stay roughly constant. Spammers effectively break localization of the observed DNS queries as spam e-mail is sent without any difference to all users around the world. Additionally, a large number of new domains are registered every day. According to Domain Tools [23], on average there are almost 1 million new domains registered per day, just in the .com, .net and .org domains. At the time when this paper was written, the number of active domains in these three top level domains was close to 70 million.

5.4 Typo squatter domains

Typo squatter domains are used mostly in phishing attacks. During the two weeks of monitoring several new typo squatter domains were detected. Of particular interest were domains related to the University of Auckland, so using the approach described in Microsoft's document [24], all permutations of the word "auckland" in the auckland.ac.nz domain where checked in the database. This test resulted in the following table of typo squatter domains related to the University of Auckland:

RR type	Query	Answer
A	aukland.ac.nz	70.85.154.28
CNAME	ec.aukland.ac.nz	aukland.ac.nz
CNAME	www.aukland.ac.nz	aukland.ac.nz
CNAME	www.health.aukland.ac.nz	aukland.ac.nz
A	www.www.aukland.ac.nz.com	202.174.119.208
A	www.webmail.aukland.ec.ac.nz.com	202.174.119.208

Table 2: *List of typo squatter domains related to the University of Auckland*

The first typo squatter domain listed in the table above is directly related to the University of Auckland. It is clear the attacker is relying on users who misspell the University of Auckland's domain, auckland.ac.nz. Malicious actions are even more obvious as the attacker created a wildcard domain aukland.ac.nz. The other typo squatter domain listed is not a real attack but more a misconfiguration of the WWW browser. In this case, the web browser added www and .com to the host name, so the complete DNS query did not end in the .nz, but in the .com top level domain. Incidentally, there exists a nz.com domain which also has a wildcard configuration so any subdomains will have proper A RR answers.

The other typo squatter domain analyzed was related to the National Bank in Auckland, New Zealand. The main web site for the National Bank is www.nationalbank.co.nz. During the analysis, a typo squatter domain was detected at www.thenationalbank.co.nz. This domain is not owned by the National Bank and, although it did not contain phishing elements (it was completely different to the real National Bank web site), it did appear to be malicious.

Further investigation of the typo squatter domain for the bank revealed a series of similar web sites located on the same physical server:

Typo squatter domain	Real domain
www.comopolitan.com	www.cosmopolitan.com
www.snestation.com	www.senstation.com
www.nationabank.co.nz	www.nationalbank.co.nz
www.playstion.com	www.playstation.com
www.wetpac.co.nz	www.westpac.co.nz

Table 3: *Partial list of typo squatter domains hosted on detected site*

The list above clearly shows typo squatter domains which were all hosted on one machine. The list above is trimmed as this particular site hosted a total of 844 domains that were collected during the two week period that the DNS traffic was passively replicated.

5.5 Fast flux DNS domains

Fast flux DNS domains are in most cases used by malicious users to control malware on infected machines. By building a replicated DNS table it is very easy detect fast flux domains and domains with unusually high number of DNS changes.

By running a query which showed domains with the highest number of changed or assigned IP addresses, besides malicious C&C servers, several anomalies were also detected. These anomalies do not appear to be malicious, but we could not explain what caused them.

The query for A resource record for the ntc.net.pk name had 1200 entries in the database. This name had an A resource record for absolutely every IP address in the 202.83.160.0 - 202.83.175.255 address space. Checking the WHOIS database [25] it was determined that the owner of this IP address space is the National Telecom Corporation in Pakistan, and that the name is legitimate, but it is difficult to explain why there are all these DNS resource records assigned.

Another similar example was noticed with the ecol.net domain, which had A resource records for 340 different IP addresses, however, these were spread among three different IP subnets, but still very close to each other.

A typical example of fast flux domains which were used as C&C servers for botnets was noticed with the swiss-invest.cn domain. This domain changed IP address 92 times in the

two weeks that the DNS traffic was monitored. IP addresses that were assigned as A resource records for this domain were scattered around the world. After issuing reverse (PTR) lookups on these IP addresses, it can be seen that the majority of the IP addresses are DSL connections on various ISPs around the world. As these ISPs are large, the customer base is large as well, which makes it easier for attackers to find machines which can be compromised and used later for malicious activities. Time differences between when these records were seen show that the TTL had to be very small, as the DNS servers repeatedly tried to resolve this domain. The table below shows the first 10 records from the database, with the time when they were first seen:

Time	**Answer**	**PTR RRs resolved later**
2006-05-18 13:53:27	68.36.69.57	c-68-36-69-57.hsd1.nj.comcast.net
2006-05-18 14:30:11	66.90.183.142	66-90-183-142.dyn.grandenetworks.net
2006-05-18 15:28:33	67.182.187.16	c-67-182-187-16.hsd1.ca.comcast.net
2006-05-18 17:14:57	68.83.57.37	c-68-83-57-37.hsd1.pa.comcast.net
2006-05-18 18:57:59	24.168.252.97	cpe-024-168-252-097.sc.res.rr.com
2006-05-19 03:47:37	24.207.139.46	24-207-139-46.dhcp.stls.mo.charter.com
2006-05-19 08:38:34	68.15.48.10	wsip-68-15-48-10.ri.ri.cox.net
2006-05-19 11:29:04	72.229.107.26	cpe-72-229-107-26.nyc.res.rr.com
2006-05-19 16:31:48	68.48.204.63	c-68-48-204-63.hsd1.va.comcast.net
2006-05-20 16:11:05	24.177.31.175	24-177-31-175.dhcp.chtn.wv.charter.com

Table 4: *Fast flux domain records*

Pointer resource record (PTR) lookups in the IN-ADDR.ARPA domain for the IP addresses that were used (PTR RRs) confirm that almost all IP addresses belong to dynamic customer connections on various ISPs.

5.6 RFC 1918 addresses

RFC 1918 defines private IP address space that can be used internally anywhere. As network traffic coming to or from these IP addresses must not be routed, there is no risk of having collisions on the Internet.

As was noted in [26], there are a substantial number of misconfigured DNS servers that have resource records with assigned IP address from the private address space. The database logged 513 resource records in 10/8 address space, 302 records in 172.16/12 address space, 693 records in 192.168/16 address space and even 115 records in the autoconfigure address 169.254/16. In total these records present 0.04% of all the records in the database.

5.7 Not recommended characters in DNS names

According to RFC 1034, a DNS name should only contain “any one of the 52 alphabetic characters A through Z in upper case and a through z in lower case”, as well as the “-“ character. The RFC recommends using names like this in order to avoid problems with some protocols like TELNET or mail. It is worth noting that although the RFC document (RFC 1034) does not recommend usage of these characters in DNS names, there is nothing preventing clients from actually using them.

The analysis of the replicated DNS database resulted in several entries which consisted of non recommended characters. Subsequent DNS queries confirmed that these hosts are indeed resolvable. Multiple DNS names using characters such as @, * and even binary, non-ascii characters were detected in the database. Example of these resource records is shown in the following table:

Query	**Type**	**Answer**
%20www.usatech.com	A	216.127.247.22
%20www.bedandbreakfastireland.net	A	82.195.130.224
www.paypal.com%20cgi-bin%20webscr%20cmd—secure-amp-sh-u%20%20.userid.jsp.krblrice.com	A	217.17.140.85
*.sharepoint.bcentral.com	A	207.68.165.26
quixta**.i8818.com	A	220.194.54.114
moll-expert.com	MX	\009mailhost.moll-expert.com
moll-expert.com	MX	\009mailbackup.moll-expert.com

Table 5: *DNS names with non recommended characters*

The intention of these names is not completely clear. As %20 (hexadecimal) presents decimal number 32, which is encoded “space” character, it is possible that most of the entries using this character are just misconfigured. However, during testing with various browsers, it was demonstrated that Internet Explorer browser will automatically encode space character in the URL to %20, no matter if the space character is part of the host name or not. The Mozilla browser did not accept host names like this. As RFC 3986 [27] endorses the use of URL encoded host names, it can be assumed that Mozilla and other browsers will also support this.

The third entry in the table, which is also using URL encoded host name, is clearly a malicious URL. This DNS entry is actually a wildcard pointer at userid.jsp.krblrice.com,

it is possible that phishers use this in order to lure users who think they are visiting Paypal's web site. At the time of writing, the phishing site did not seem to be functional, although the DNS name could still be resolved.

The host name *.sharepoint.bcentral.com clearly presents a misconfiguration as the administrator probably wanted to define a wildcard DNS host name.

Finally, moll-expert.com mail exchange resource records have two names starting with a binary character, 0x09. This was probably a result of the editor program the administrator used to edit the zone file.

6. Conclusions and future work

Passive DNS replication allows analysis of DNS data that could not have been performed otherwise. By building a stable database that is regularly updated it is possible to detect anomalies. If both queries and replies are logged and correlated, they can lead to detection of infected or otherwise malicious machines on the local network. The DNS can in this case be used as part of an Intrusion Detection System (IDS).

Analysis of logged data showed a high number of misconfigured DNS servers or those that are serving incorrect or internal data.

Future work includes correlation of queries and replies, which will make sure that the deployment is resistant to poisoning attacks, at least from the passive DNS replication point of view. DNS extensions should be properly parsed so the data can be entered into the database and also an interface to the WHOIS database should be provided. DNSSEC [28] is also expected to be more in use in the future, so it can be used to verify logged DNS replies.

Analysis of data in the database should be automated as much as possible to identify abuses and anomalies. Data logged in the database can be correlated with Microsoft's Strider URL Tracer software in order to detect typo squatter domains. Internationalized Domain Names (IDN) [29] allow domain names in non-ASCII characters. Such records should be also analyzed for potential typo squatter or different attacks.

Additionally, it would be interesting to record number of new domains seen in the DNS queries for a longer period of time to see if it will decrease with the time.

7. Acknowledgments

I would like to thank Brett Lomas and Nevil Brownlee for their useful ideas during this project.

8. References

[1] P. V. Mockapetris, K. J. Dunlap, "Development of the Domain Name System," ACM Symphosium proceedings on Communications architectures and protocols (SIGCOMM '88), vol. 18, issue 4, 1998.

[2] P. Mockapetris, "Domain Names – Concepts and Facilities," RFC 1034, November 1987.

[3] P. Mockapetris, "Domain Names – Implementation and Specification," RFC 1035, November 1987.

[4] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, T. Berners-Lee, "Hypertext Transfer Protocol – HTTP/1.1," RFC 2616, June 1999.

[5] Gadi Evron, SecuriTeam Blog, "Looking behind the smoke screen of the Internet: DNS recursive attacks, spamvertised domains, phishing, botnet C&C's, International Infrastructure and you," http://blogs.securiteam.com/index.php/archives/298.

[6] M. Wong, W. Schlitt, "Sender Policy Framework (SPF) for Authorizing Use of Domains in E-Mail, Version 1," RFC 4408, April 2006.

[7] LURHQ Threat Intelligence Group, "DNS Cache Poisoning – The Next Generation," http://www.lurhq.com/cachepoisoning.html.

[8] M. Zalewski, "Strange Attractors and TCP/IP Sequence Number Analysis," http://www.bindview.com/Services/Razor/Papers/2001/tcpseq.cfm, 2001.

[9] F. Weimer, "Passive DNS Replication," FIRST 2005, April 2005.

[10] A. Schonewille, D. v. Helmond, "The Domain Name Service as an IDS", Research Project for the Master System- and Network Engineering at the University of Amsterdam, February 2006.

[11] J. Kristoff, "DNSWatch", http://aharp.ittns.northwestern.edu/software/dnswatch, September 2004.

[12] N. Elton, M. Keel, "A Discussion of Bot Networks," EDUCAUSE 2005, http://www.educause.edu/ir/library/pdf/SPC0568.pdf, April 2005.

[13] K. Ishibashi, T. Toyono, K. Toyama, M. Ishino, H. Ohshima, I. Mizukoshi, "Detecting Mass-Mailing Worm Infected Hosts by Mining DNS Traffic Data," ACM Symphosium proceedings on Communications architectures and protocols (SIGCOMM '05), pp 159-164, August 2005.

[14] VeriSign, "COM NET Registry, TLD Zone Access Program," http://www.verisign.com/information-services/naming-services/com-net-registry/page_001052.html.

[15] BIND website, http://www.isc.org/products/BIND/.

[16] Tcpdpriv, "A program for eliminating confidential information from packets collected on a network interface," http://ita.ee.lbl.gov/html/contrib/tcpdpriv.html, October 2005.

[17] Internet Engineering Task Force, "Requirements for Internet Hosts – Application and Support," RFC 1123, October 1989.

[18] P. Vixie, "Extension Mechanisms for DNS (EDNS0)," RFC 2671, August 1999.

[19] Tcpdump website, http://www.tcpdump.org/, October 2005.

[20] SpamAssassin website, http://spamassassin.apache.org, May 2006.

[21] P. Vixie, "MAPS RBL Rationale," July 2000.

[22] J. Jung, E. Sit, H. Balakrishnan, R. Morris, "DNS Performance and the Effectiveness of Caching," ACM Transactions on Networking, Vol. 10, No. 5, pp. 589-603, October 2002.

[23] Domain Tools Internet Statistics website, http://www.domaintools.com/internet-statistics/.

[24] Y. Wang, D. Beck, J. Wang, C. Verbowski, B. Daniels, "Strider Typo-Patrol: Discovery and Analysis of Systematic Typo-Squatting," Microsoft Research Technical Report (to be submitted to the 2nd Usenix Workshop on Steps to Reducing Unwanted Traffic on the Internet (SRUTI '06)), http://research.microsoft.com/URLTracer/.

[25] L. Daigle, "WHOIS Protocol Specification," RFC 3912, September 2004.

[26] N. Brownlee, kc Claffy, E. Nemeth, “DNS Measurements at a Root Server,” Global Telecommunications Conference, 2001 (GLOBECOM ’01), IEEE, Volume 3, pp 1672 – 1676, November 2001.

[27] T. Berners-Lee, R. Fielding, L. Masinter, “Uniform Resource Identifier (URI): Generic Syntax,” RFC 3986, January 2005.

[28] R. Arends, R. Austein, M. Larson, D. Massey, S. Rose, “DNS Security Introduction and Requirements,” RFC 4033, March 2005.

[29] P. Hoffman, M. Blanchet, “Preparation of Internationalized Strings (“stringprep”)”, RFC 3454, December 2002.