BRNO UNIVERSITY OF TECHNOLOGY
Faculty of Information Technology

NETWORK APPLICATIONS AND
NETWORK ADMINISTRATION
2018/2019

# EXPORT OF DNS INFORMATION USING SYSLOG PROTOCOL

Tomáš Daniš (xdanis05) Brno, 18.11 2018

# Contents

# 1 Introduction

In this project we implement a tool for extracting information from DNS messages and collecting statistics about this information. The application can capture all packets passing through a specified network interface or read already captured packets from a file. It filters out DNS responses and extracts DNS records the server has responded with. The application keeps statistics about these records. There is also a possibility to regularly report these statistics to a Syslog server.

# 2 Theoretical foundations

This section presents the concepts underlying the functionality of dns-export.

## 2.1 Domain Name System (DNS)

DNS is a network service used mainly for translating domain names into network addresses. In practice, users want to be able to access network services using easy-to-remember names instead of complex network addresses. To provide this functionality, DNS was created. Any user providing a service on the internet can register a domain name for their network address, making their service easily accessible by other users. When a user wants to access a network resource using a domain name, he contacts a DNS server, which translates the given domain name into a network address and sends it back to the user. This address then can be used to connect to the requested service.

While domain name translation is the main functionality of DNS, it has found use in other areas as well. DNS is also used for locating specific services in the network or mail server identification when forwarding e-mails.

### 2.1.1 Protocol operation

We will now examine the operation of the DNS protocol. Figure 1 illustrates the communication of a DNS client and DNS servers.

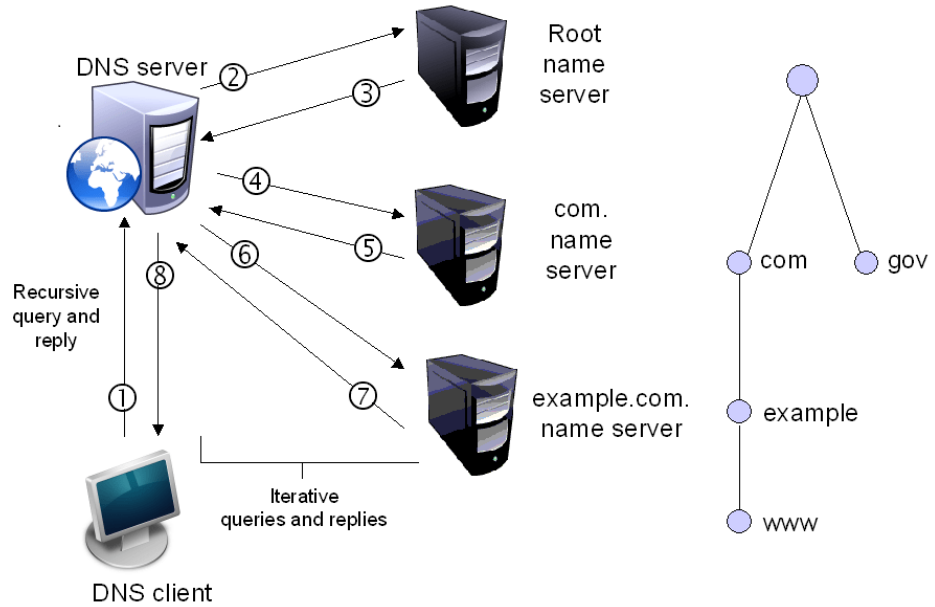


**Figure 1:** DNS Operation, source: https://securityaffairs.co/wordpress/3184/cyber-crime/anonymous-dns-amplification-attacks-for-operation-global-blackout.html

At the beginning, a client wants to translate the domain name “example.com” into a network address. It therefore sends a *DNS query* to its configured DNS server (step 1). A single DNS server cannot hold the translations for the entire internet. The DNS server therefore asks other DNS servers for information about „example.com“. When a server doesn’t know the answer, it responds with information about another DNS server which to contact next. Without going into too much detail, eventually a server which knows the translation for “example.com” is found (steps 2 – 7). Our DNS server requests this information and receives a *DNS response*, which is then forwarded back to the DNS client (step 8).

### 2.1.2 Message format

A core functionality necessary in this project is being able to read and interpret DNS responses. That is why we will now look at the format of a DNS response message and its contents. Figure 2 illustrates the format of a DNS message.

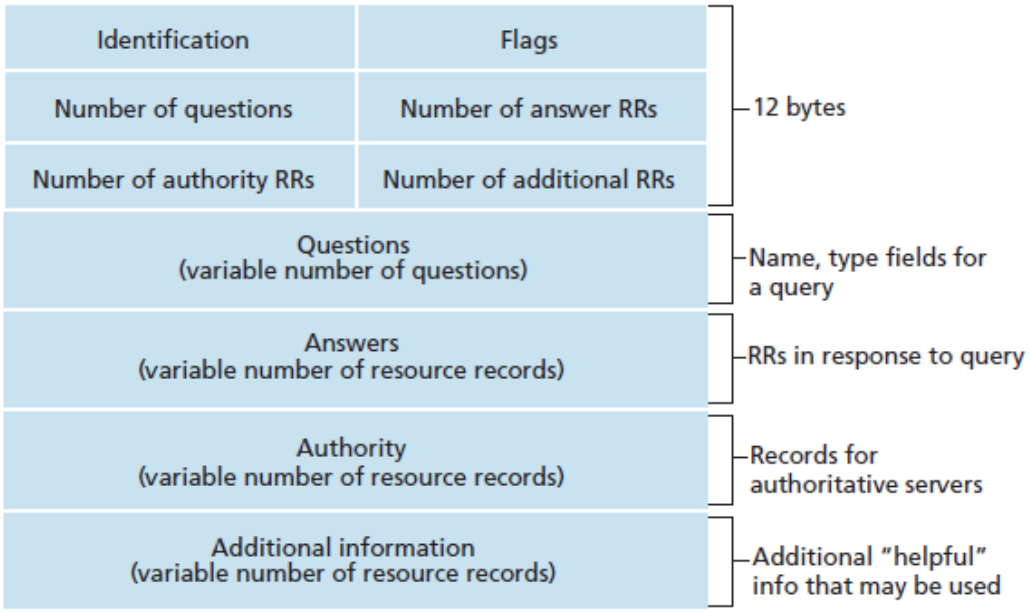


**Figure 2:** DNS message format, source: https://electronicspost.com/dns-messages/

The fields of the message are as follows:

- **Identification** – a 2-byte number identifying the communication. A DNS query and a DNS response share this value to make it easy to determine which query does a response belong to
- **Flags** – Bitmap used for communicating server and message settings
- **Number of questions** – Number of queries in the “Questions” section
- **Number of answer RRs** – Number of DNS resource records in the “Answers” section
- **Number of authority RRs** – Number of DNS resource records in the “Authority” section
- **Number of additional RRs** – Number of DNS resource records in the “Additional information” section
- **Questions** – Variable number of queries in the message. In theory, it is possible to send multiple DNS queries in a single message.
- **Answers** – Variable number of answer resource records (RRs). Each answer RR contains information sent by the DNS server.

- **Authority** – Variable number of answer RRs. This section contains information about other DNS servers when it is needed.
- **Additional information** – Variable number of answer RRs. Contains additional potentially useful information.

dns-export extracts information from the "Answers" section, which contains answer RRs. Therefore, we will next look at the format of a single answer RR. Figure 3 illustrates the format of an answer RR.

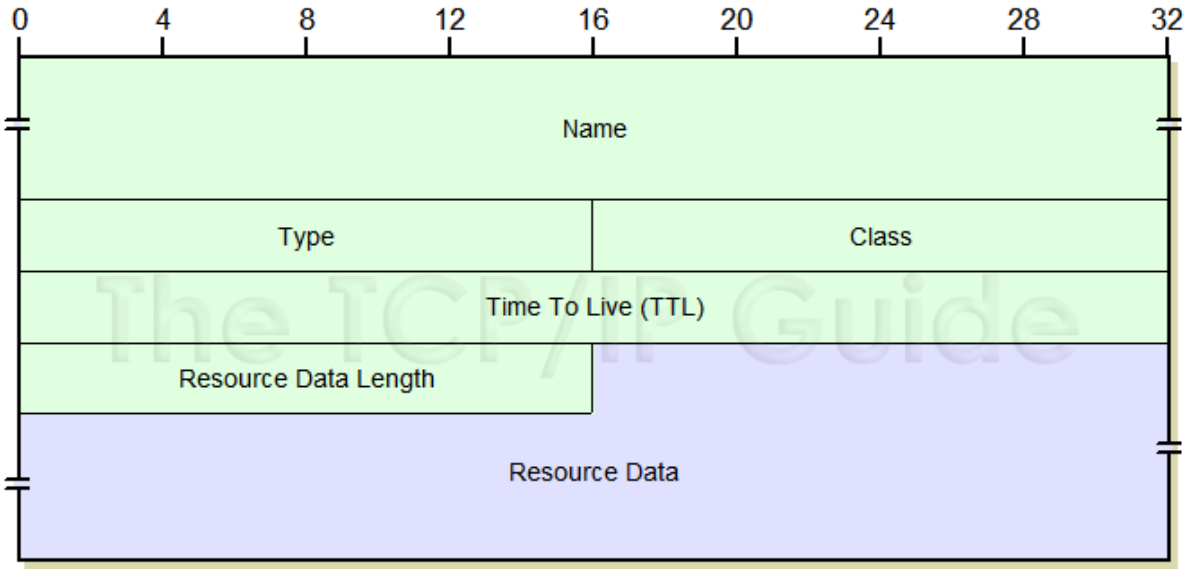


**Figure 3:** Answer RR format, source: http://www.tcpipguide.com/free/t_DNSMessageResourceRecordFieldFormats-2.htm

The fields are as follows:

- **Name –** Contains the domain name which the answer RR holds information for.
- **Type –** Type of the RR. Specifies what kind of data this RR contains.
- **Class –** Class of the resource record, normally 1 for Internet
- **Time To Live (TTL) –** Specifies the number of seconds the RR should be retained in cache
- **Resource Data Length –** Specifies the length of the next section "Resource Data".
- **Resource Data –** Contains the information from the DNS server itself. What data is found here depends on the "Type" field. For example, if the type field is 1, the RR type is type "A" which is used for translating domain names into IPv4 addresses. In this case, this field contains the 32bit IPv4 address corresponding to the domain name in the "Name" field.

# 3 Application design

The application is split into modules, with each providing a part of the overall functionality. These modules are as follows:

- **dns-export –** The main module of the application. Parses arguments and calls on *sniffer* to analyze the DNS traffic from the specified source.
- **sniffer –** Module for capturing packets on an interface or reading them from a file. Captured packets are processed and analyzed.

- **headers –** Module for extracting application layer data from captured packets. This module can parse ISO OSI L2-L4 headers and strip them from the message. It is also responsible for reconstructing the message from multiple IP fragmented packets or multiple TCP segments.
- **message_reassembler –** Module providing the functionality of reconstructing the original message out of multiple fragmented/segmented messages which may arrive out of order. This module is used by the *headers* module to reconstruct the complete DNS messages.
- **dns_parser –** Module for parsing DNS messages and extracting information from them. This module reads the RRs in the Answers section of the DNS message, interprets them and constructs a string representation of each RR in accordance with the respective RFCs which define the RRs.
- **base64 –** Module for working with data in Base64 format. Base64 is used in some RRs to encode sent data and this module is used to parse it.
- **stats_report –** Module for the collected statistics. Contains methods for sending the statistics to various destinations. In our case, either a Syslog server or the standard output.

Figure 4 illustrates a high-level overview of the data flow within the application.

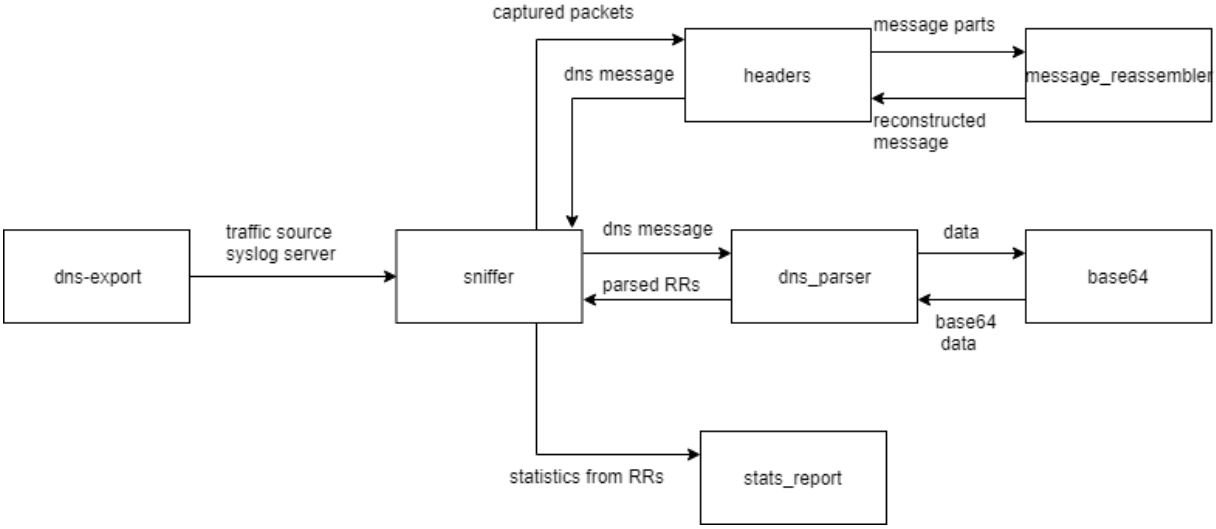


**Figure 4:** dns-export data flow

# 4 Examples & Usage

The application is run using the following command:

*dns-export [-r FILE | -i IFACE ] [-s HOST ] [ -t SECONDS ]*

where the meaning of the options is the following:

- -r - PCAP source file to read packets from
- -i – Interface to capture traffic on.
- -s – Syslog server to export statistics to
- -t - Syslog statistics export period

**Examples:**
*dns-export -i eth0 -s syslog.example.com -t 30*
The application captures packets on the eth0 interface and sends calculated statistics to a syslog server syslog.example.com every 30 seconds.

*dns-export -r dns.pcap -s syslog.example.com*
Processes DNS messages in dns.pcap file and exports calculated statistics to the syslog.example.com syslog server and exits.

# 5 Conclusion

In this project, we have implemented a tool for analyzing DNS traffic and exporting statistics about it. The required technologies turned out to come from a variety of programming fields, ranging from low-level network programming, to data encoding/decoding. The end result is a tool capable of analyzing a great majority of DNS communication and is ready for deployment in network monitoring systems.

# 6 Sources

RFC1035
RFC4034
RFC3548
RFC3596
RFC5424
RFC3339