|  |  |  |
| --- | --- | --- |
| **МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ**  **РОССИЙСКОЙ ФЕДЕРАЦИИ** | | |
| ФГБОУ ВПО  **«БРЯНСКИЙ ГОСУДАРСТВЕННЫЙ**  **ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»** | | |
| **Кафедра «**Информатика и программное обеспечение**»** | | |
|  | |  |
| **КУРСОВАЯ РАБОТА ПО ДИСЦИПЛИНЕ** | | |
| **«СЕТИ ЭВМ И ТЕЛЕКОММУНИКАЦИИ»** | | |
|  | | |
| «ИЗУЧЕНИЕ СЕТЕВОГО ПРОТОКОЛА DNS (TCP/UDP 53)» | | |
|  | Выполнил студ. гр. З16-ИВТ, зач. кн. №16.0649 | |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Мамаков А.В. | |
|  | «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2017 г. | |
|  | Руководитель | |
|  | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ к.т.н., доц. Коростелёв Д.А. | |
|  | «\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2017 г. | |
| Брянск 2017 | | |

ЗАДАНИЕ

1. Изучить назначение и формат пакетов протокола IP DNS (TCP/UDP 53).
2. С помощью сниффера (например, Wireshark) захватить несколько пакетов этого протокола. При необходимости установить программное обеспечение, необходимое для генерации пакетов указанного протокола. Приложить к отчету файл с захваченными пакетами.
3. Исследовать поля протокола IP на примере нескольких пакетов.
4. Написать программу для захвата и расшифровки полей пакетов протокола DNS (TCP/UDP 53).

Аннотация

На основании здания курсовой работы был произведен захват и анализ содержимого сетевых пакетов протокола DNS при помощи приложения Wireshark, а также разработано приложение для захвата и расшифровки полей пакетов протокола DNS. В процессе анализа и разработки была изучена литература и документация по сетевым протоколам TCP, UDP и DNS. Рассмотрены особенности функционирования протоколов, формат заголовков, функции, выполняемые полями заголовков. Разработано приложение – анализатор пакетов протокола DNS, позволяющее производить захват и анализ полей заголовков IP-пакетов, UDP-дейтаграмм, TCP-сегментов и DNS запросов.

# Содержание

[Содержание 4](#_Toc503946846)

[Введение 5](#_Toc503946847)

[1. АНАЛИТИЧЕСКАЯ ЧАСТЬ 7](#_Toc503946848)

[1.1. Описание протокола TCP 7](#_Toc503946849)

[1.2. Описание протокола UDP 9](#_Toc503946850)

[1.3. Описание протокола DNS 10](#_Toc503946851)

[2. КОНСТРУКТОРСКАЯ ЧАСТЬ 16](#_Toc503946852)

[2.1. Анализ содержимого сетевых пакетов DNS 16](#_Toc503946853)

[2.2. Описание приложения 17](#_Toc503946854)

[3. ЛИСТИНГ ПРОГРАММЫ 18](#_Toc503946855)

[Заключение 19](#_Toc503946856)

[Список литературы 20](#_Toc503946857)

# Введение

Стек протоколов TCP/IP - это организованный набор основных сетевых протоколов, который иерархическим способом разделен на четыре уровня и представляет собой систему транспортного распределения пакетов по компьютерной сети. TCP/IP - это наиболее известный стек сетевых протоколов, который используется на данный момент. Стек протоколов TCP/IP предусматривает четыре уровня:

- прикладной уровень (application layer),

- транспортный уровень (transport layer),

- сетевой уровень (Internet layer),

- канальный уровень (link layer).

На прикладном уровне работает большинство сетевых приложений. Эти программы имеют свои собственные протоколы обмена информацией, например, интернет браузер для протокола HTTP, ftp-клиент для протокола FTP, почтовая программа для протокола SMTP (электронная почта), а также DNS (преобразование символьных имён в IP-адреса) и многие другие.

В основном все эти протоколы работают поверх TCP или UDP и привязаны к определённому порту, например: HTTP на TCP-порт 80 или 8080, FTP на TCP-порт 20 для передачи данных и 21 для управляющих команд, запросы DNS на порт UDP (реже TCP) 53 и т.д. Эти порты определены Агентством по выделению имен и уникальных параметров протоколов (IANA).

DNS протокол (от английского Domain Name System – система доменных имен) – это компьютерная четко распределенная система для получения информации о состоянии домена. Зачастую его используют для получения IP-адреса по имени определенного хоста (устройства или компьютера), получения необходимой информации о пройденном маршруте почты, которая обслуживается узлами для протоколов в домене.

Такая распределенная база данных системы доменных имен поддерживается благодаря иерархии DNS-серверов, которые взаимодействуют согласно определенному протоколу.

Основой для DNS протокола служит представление об ее иерархической системе доменного адреса и зонах. Каждый отдельный сервер, которые в ответе за имя, может вполне делегировать свою ответственность на будущую часть домена другому серверу, что способствует возложению ответственности за популярность информации на серверы разнообразных предприятий, организация или людей, которые отвечают только за собственную часть доменного имени.

# АНАЛИТИЧЕСКАЯ ЧАСТЬ

## Описание протокола TCP

Протокол TCP использует надежную связь, для обеспечения которой протокол уровня 4 посылает подтверждения о получении данных и запрашивает повторную передачу, если данные не получены или искажены. TCP используется в таких прикладных протоколах, как HTTP, FTP, SMTP и Telnet.

Протокол TCP требует, чтобы перед отправкой сообщения было открыто соединение. Серверное приложение должно выполнить так называемое пассивное открытие (passive open), чтобы создать соединение с известным номером порта, и, вместо того чтобы отправлять вызов в сеть, сервер переходит в ожидание поступления входящих запросов. Клиентское приложение должно выполнить активное открытие (active open), отправив серверному приложению синхронизирующий порядковый номер (SYN), идентифицирующий соединение. Клиентское приложение может использовать динамический номер порта в качестве локального порта.

Сервер должен отправить клиенту подтверждение (ACK) вместе с порядковым номером (SYN) сервера. В свою очередь клиент отвечает АСК, и соединение устанавливается. После этого может начаться процесс отправки и получения сообщений. При получении сообщения в ответ всегда отправляется сообщение АСК. Если до получения АСК отправителем истекает тайм-аут, сообщение помещается в очередь на повторную передачу.

Поля заголовка TCP перечислены в следующей таблице:

|  |  |  |
| --- | --- | --- |
| Поле | Длина | Описание |
| Порт источника | 2 байта | Номер порта источника |
| Порт назначения | 2 байта | Номер порта назначения |
| Последовательный номер | 4 байта | Последовательный номер генерируется источником и используется назначением, чтобы переупорядочить пакеты для создания исходного сообщения и отправить подтверждение источнику. |
| Номер подтверждения | 4 байта | Если установлен бит АСК поля "Управление", в данном поле содержится следующий ожидаемый последовательный номер. |
| Смещение данных | 4 бита | Информация о начале пакета данных. |
| Резерв | 6 бит | Резервируются для будущего использования. |
| Управление | 6 бит | Биты управления содержат флаги, указывающие, верны ли поля подтверждения (АСК), указателя срочности (URG), следует ли сбрасывать соединение (RST), послан ли синхронизирующий последовательный номер (SYN) и т. д. |
| Размер окна | 2 байта | В этом поле указывается размер приемного буфера. Используя подтверждающие сообщения, получатель может информировать отправителя о максимальном размере данных, которые тот может отправить. |
| Контрольная сумма | 2 байта | Контрольная сумма заголовка и данных; по ней определяется, был ли искажен пакет. |
| Указатель срочности | 2 байта | В этом поле целевое устройство получает информацию о срочности данных. |
| Опции | переменная | Необязательные значения, которые указываются при необходимости. |
| Дополнение | переменная | В поле дополнения добавляется столько нулей, чтобы заголовок заканчивался на 32-битной границе. |

Табл. 1 Поля заголовка TCP

TCP — это сложный, требующий больших затрат времени протокол, что объясняется его механизмом установления соединения, но он берет на себя заботу о гарантированной доставке пакетов, избавляя нас от необходимости включать эту функциональную возможность в прикладной протокол.

Протокол TCP имеет встроенную возможность надежной доставки. Если сообщение не отправлено корректно, мы получим сообщение об ошибке. Протокол TCP определен в RFC 793.

## Описание протокола UDP

В отличие от TCP UDP — очень быстрый протокол, поскольку в нем определен самый минимальный механизм, необходимый для передачи данных. Сообщения поступают в любом порядке, и то, которое отправлено первым, может быть получено последним. Доставка сообщений UDP вовсе не гарантируется, сообщение может потеряться, и могут быть получены две копии одного и того же сообщения. Последний случай возникает, если для отправки сообщений в один адрес использовать два разных маршрута.

UDP не требует открывать соединение, и данные могут быть отправлены сразу же, как только они подготовлены. UDP не отправляет подтверждающие сообщения, поэтому данные могут быть получены или потеряны. Если при использовании UDP требуется надежная передача данных, ее следует реализовать в протоколе более высокого уровня.

В рамках использования протокола UDP нужно различать однонаправленную передачу, широковещательную передачу и групповую рассылку.

Однонаправленное (unicast) сообщение отправляется из одного узла только в один другой узел. Это также называется связью "точка-точка". Протокол TCP поддерживает лишь однонаправленную связь. Если серверу нужно с помощью TCP взаимодействовать с несколькими клиентами, каждый клиент должен установить соединение, поскольку сообщения могут отправляться только одиночным узлам. Широковещательная передача (broadcast) означает, что сообщение отправляется всем узлам сети. Групповая рассылка (multicast) - это промежуточный механизм: сообщения отправляются выбранным группам узлов.

UDP может использоваться для однонаправленной связи, если требуется быстрая передача, например для доставки мультимедийных данных, но главные преимущества UDP касаются широковещательной передачи и групповой рассылки.

Обычно, когда мы отправляем широковещательные или групповые сообщения, не нужно получать подтверждения из каждого узла, поскольку тогда сервер будет наводнен подтверждениями, а загрузка сети возрастет слишком сильно. Примером широковещательной передачи является служба времени. Сервер времени отправляет широковещательное сообщение, содержащее текущее время, и любой хост, если пожелает, может синхронизировать свое время с временем из широковещательного сообщения.

Заголовок UDP гораздо короче и проще заголовка TCP:

|  |  |  |
| --- | --- | --- |
| Поле | Длина | Описание |
| Порт источника | 2 байта | Указание порта источника для UDP необязательно. Если это поле используется, получатель может отправить ответ этому порту. |
| Порт назначения | 2 байта | Номер порта назначения |
| Длина | 2 байта | Длина сообщения, включая заголовок и данные. |
| Контрольная сумма | 2 байта | Контрольная сумма заголовка и данных для проверки |

Табл. 2. Поля заголовка UDP

UDP — это быстрый протокол, не гарантирующий доставки. Если требуется поддержание порядка сообщений и надежная доставка, нужно использовать TCP. UDP главным образом предназначен для широковещательной и групповой передачи. Протокол UDP определен в RFC 786.

## Описание протокола DNS

DNS (англ. Domain Name System — система доменных имён) — компьютерная распределённая система для получения информации о доменах. Чаще всего используется для получения IP-адреса по имени хоста (компьютера или устройства), получения информации о маршрутизации почты, обслуживающих узлах для протоколов в домене (SRV-запись). Распределённая база данных DNS поддерживается с помощью иерархии DNS-серверов, взаимодействующих по определённому протоколу.

Существует два вида запросов: рекурсивные и итеративные. Первый вид предполагает получение клиентом IP-адреса, а второй - адреса сервера, который может сообщить адрес. В одном запросе может содержаться несколько вопросов. Если сервер не сможет ответить на вопросы, он пришлет отклик, где содержатся адреса других серверов, способных решить эту задачу. Ниже на рис. 1 представлен формат таких сообщений (в качестве транспорта используется UDP или TCP, порт 53).

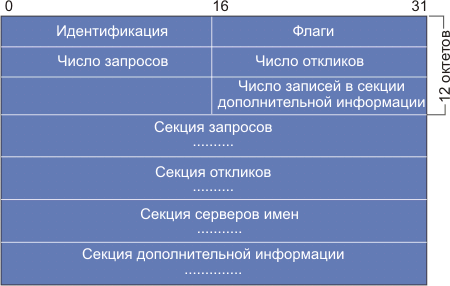


Рис. 1 Формат DNS-сообщений

Каждое сообщение начинается с заголовка, который содержит поле идентификация, позволяющее связать в пару запрос и отклик. Поле флаги определяет характер запрашиваемой процедуры, а также кодировку отклика. Поле число запросов определяет число записей соответствующего типа, содержащихся в сообщении. Так число запросов задает число записей в секции запросов, где записаны запросы, требующие ответов. Каждый вопрос состоит из символьного имени домена, за которым следует тип запроса и класс запроса. Значения битов поля флаги в сообщении сервера имен отображены в таблице 3. Разряды пронумерованы слева направо, начиная с нуля – см. рис. 2.

http://citforum.ru/nets/semenov/4/44/dns_4.gif

Рис. 2

|  |  |
| --- | --- |
| Код поля *флаги* | Описание |
| 0 (QR) | Операция:  0 Запрос  1 Отклик |
| 1…4 | Тип запроса (opcode):  0 стандартный  1 инверсный  2 запрос состояния сервера |
| 5 (AA) | Равен 1 при отклике от сервера (RR), в ведении которого находится домен, упомянутый в запросе. |
| 6 (TC) | Равен 1 при укорочении сообщения. Для UDP это означает, что ответ содержал более 512 октетов, но прислано только первые 512. |
| 7 (RD) | Равен 1, если для получения ответа желательна рекурсия. |
| 8 (RA) | Равен 1, если рекурсия для запрашиваемого сервера доступна. |
| 9…11 | Зарезервировано на будущее. Должны равняться нулю. |
| 12…15 | Тип отклика (rcode):  0   нет ошибки  1   ошибка в формате запроса  2   сбой в сервере  3   имени не существует |

Табл. 3 Значения битов поля флаги

Ниже описан формат секции запросов в DNS-сообщении.

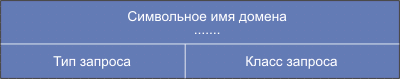


Рис. 4. Формат секции вопросов DNS-запроса.

Поле символьное имя домена имеет переменную длину, содержит одно или более субполей, начинающихся с байта длины (0-63). Поле завершается 0. Например, для ns.itep.ru (цифры представляют собой октеты длины):

http://citforum.ru/nets/semenov/4/44/dns_6.gif

Рис. 5.

В реальной нотации байты длины субполя могут иметь два старших бита равные 1, что преобразует интервал значений из 0-63 в 192-255. Такой байт в поле означает, что это не мера длины секции, а 16-битный указатель, 14 бит которого являются смещением от начала DNS-сообщения, указывающим на место продолжения секции. Смещение для первого байта поля идентификации равно нулю. Эти ухищрения придуманы для сокращения длины сообщений, так как одно и то же имя домена в отклике может повторяться много раз. Поле тип запроса характеризует разновидность запроса:

|  |  |  |
| --- | --- | --- |
| Тип запроса | Код запроса | Описание |
| A | 1 | IP-адрес |
| NS | 2 | Сервер имен. |
| CNAME | 5 | Каноническое имя. |
| SOA | 6 | Начало списка серверов. Большое число полей, определяющих часть иерархии имен, которую использует сервер. |
| MB | 7 | Имя домена почтового ящика. |
| WKS | 11 | well-known service - стандартная услуга. |
| PTR | 12 | Запись указателя. |
| HINFO | 13 | Информация об ЭВМ. |
| MINFO | 14 | Информация о почтовом ящике или списке почтовых адресов. |
| MX | 15 | Запись о почтовом сервере. |
| TXT | 16 | Не интерпретируемая строка ASCII символов. |
| AXFR | 252 | Запрос зонного обмена |
| \* или ANY | 255 | Запрос всех записей. |

Табл. 3 Разновидности полей *тип запроса* и их коды

Последние две строки в табл. 3 относятся только к запросам. Поле класс запроса позволяет использовать имена доменов для произвольных объектов (все официальные имена Интернет относятся к одному классу [IN] - 1). В сообщении DNS-сервера каждая из секций (дополнительной информации, ответов или узловых серверов имен) состоит из набора ресурсных записей, которые описывают имена доменов и некоторую другую информацию (например, их адреса). Появление ресурсных полей в DNS базе данных придают ей новые качества. Каждая запись описывает только одно имя, формат этих записей отображен на рис. 4.4.12.6.

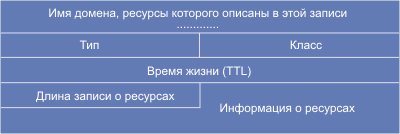


Рис.6. Формат ресурсных записей в DNS (RR)

Всего существует 20 различных типов RR-записей. Имя домена в такой записи может иметь произвольную длину. Поля тип и класс характеризуют тип и класс данных, включенных в запись (аналогичны используемым в запросах). Поле время жизни (TTL) содержит время (в секундах), в течение которого запись о ресурсах может храниться в буферной памяти (в кэше). Обычно это время соответствует двум дням. Формат информации о ресурсах зависит от кода в поле тип, так для тип=1 - это 4 байта IP-адреса. Сервер имен может обслуживать и другие запросы, например, по IP-адресу определять символьное имя домена или преобразовать имя домена в адрес почтового сервера. Когда организация присоединяется к Интернет, она получает в свое распоряжение не только определенную DNS-область, но и часть пространства в in-addr.arpa, соответствующую ее IP-адресам. Домен in-addr.arpa предназначен для определения имен по их IP-адресам. Такая схема исключает процесс перебора серверов при подобном преобразовании.

Имена в домене IN-ADDR.ARPA могут иметь до четырех субполей помимо суффикса IN-ADDR.ARPA. Каждое субполе представляет собой октет IP-адреса, и содержит последовательность символов, отображающую коды в диапазоне 0-255. Так имя для IP-адреса 192.148.166.137 (если оно существует) содержится в домене с именем 137.166.148.192.IN-ADDR.ARPA. Запись адреса задом наперед диктуется общими принципами записи имен доменов (корневая часть имени находится справа). Направив несколько запросов в домен IN-ADDR.ARPA относительно имен объектов с интересующими вас IP-адресами, можно получить следующий результат:

IP\_address Hard-addr Delay Date Host\_name

128.141.202.101 00.00.0c.02.69.7d 440 10/10/95 na48-1.cern.ch

192.148.166.102 00.00.a7.14.41.c2 5 10/10/95

192.148.166.237 00.00.0c.02.69.7d 5 10/10/95 ITEP-M9.Relcom.ITEP.RU

где в левой колонке записаны IP-адреса, имена которых ищутся, во второй - записаны аппаратные адреса интерфейсов, через которые доступны искомые объекты. Поскольку первая и третья строки относятся к внешним по отношению к узлу ITEP объектам, здесь записан адрес интерфейса пограничного маршрутизатора. В третьей колонке содержится значение RTT в мсек, а в последней - имена объектов. IP-адресу 192.148.166.102 не соответствует никакого имени.

# КОНСТРУКТОРСКАЯ ЧАСТЬ

## Анализ содержимого сетевых пакетов DNS

С помощью программы – анализатора трафика для компьютерных сетей Wireshark произведем захват сетевых пакетов протокола DNS, используя фильтрацию пакетов с портом источника и назначения равным 53.

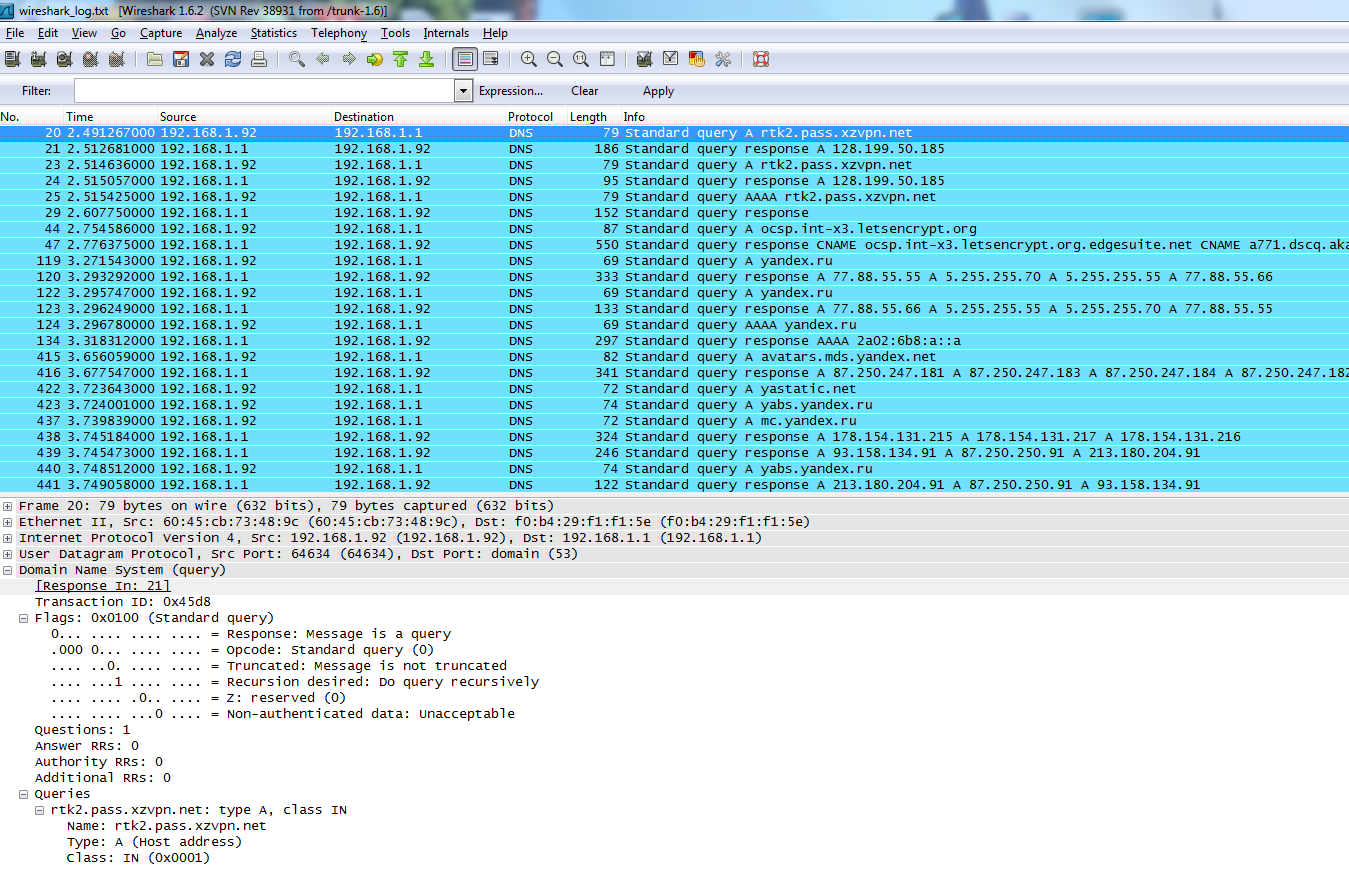


Рис.7 Захват сетевого пакета с запросом DNS

На рис. 7 представлен один из перехваченных пакетов с запросом DNS. Анализируя содержимое пакета можно отметить, что при этом использовался протокол UDP, запрос DNS содержит номер транзакции, флаги запроса, количество запросов – один, запрос идет на разрешение сетевого адреса доменного имени rtk2.pass.xzvpn.net. Ниже на рис. 8 представлен фрагмент пакета с ответом на данный запрос DNS. На скриншоте видно, что пакет с ответом имеет тот же номер транзакции, и содержит несколько ответов.

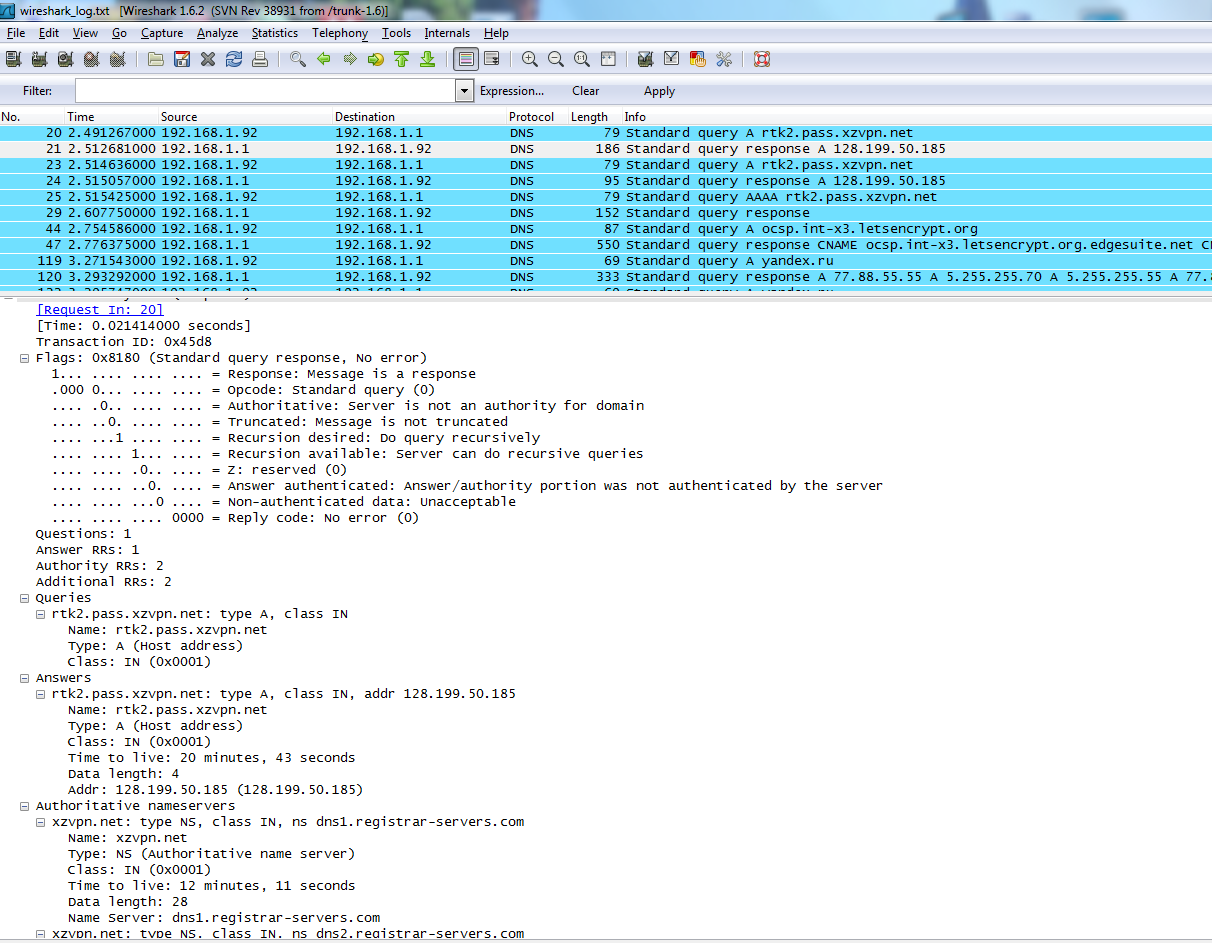
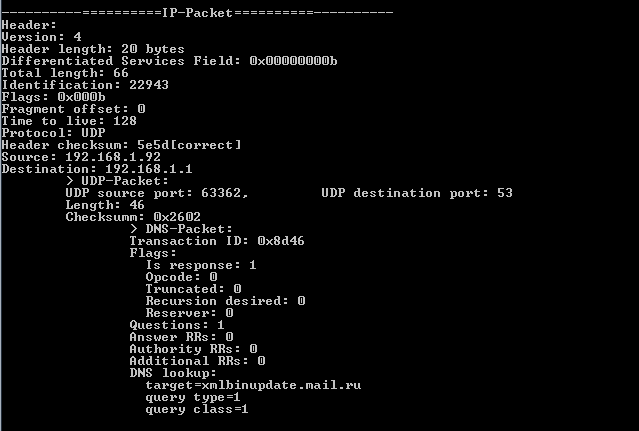


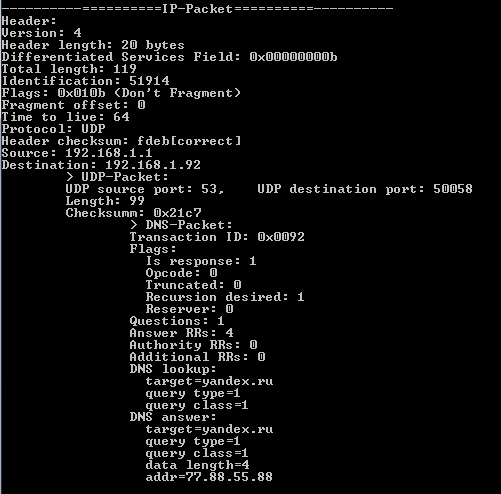
Рис.8 Захват сетевого пакета с ответом DNS

Первый ответ содержит запрошенное доменное имя, тип и класс запроса, TTL ответа (20 мин. 43 сек.) и сетевой адрес запрошенного доменного имени – 128.199.50.185. Последующие ответы содержат информацию о DNS – серверах.

## Описание приложения

В рамках данной курсовой работы разработано приложение для захвата и анализа сетевых пакетов протокола DNS на языке программирования C++ с использованием интегрированной среды разработки MS Visual Studio 2015. В приложении захват пакетов осуществляется при помощи библиотеки winsock2, в цикле захватываются IP пакеты и, если они содержат UDP или TCP с портом источника или адресата равным 53, то начинается анализ их содержимого и результат выводится на экран.





# ЛИСТИНГ ПРОГРАММЫ

В листинге

#include <conio.h>

#include <stdio.h>

#include <winsock2.h>

#include <string.h>

#define MAX\_PACKET\_SIZE 0x10000

#define MAX\_PACKET\_SIZE 0x10000

#define SIO\_RCVALL 0x98000001

#define LIL\_ENDIAN 0

#define BIG\_ENDIAN 1

#define BYTE\_ORDER BIG\_ENDIAN

#define IP\_HDR\_LEN 20

#define TCP\_HDR\_LEN 20

#define UDP\_HDR\_LEN 8

#define DNS\_HDR\_LEN 12

// Буфер для приёма данных

char Buffer[MAX\_PACKET\_SIZE]; // 64 Kb#define

typedef struct IPHeader {

UCHAR iph\_verlen; // версия и длина заголовка

UCHAR iph\_tos; // тип сервиса

USHORT iph\_length; // длина всего пакета

USHORT iph\_id; // Идентификация

USHORT iph\_offset; // флаги и смещения

UCHAR iph\_ttl; // время жизни пакета

UCHAR iph\_protocol; // протокол

USHORT iph\_xsum; // контрольная сумма

ULONG iph\_src; // IP-адрес отправителя

ULONG iph\_dest; // IP-адрес назначения

} IPHeader;

typedef struct TCPHeader

{

USHORT source\_port; // (16 bits)

USHORT destination\_port; // (16 bits)

ULONG seq\_number; // Sequence Number (32 bits)

ULONG ack\_number; // Acknowledgment Number (32 bits)

USHORT info\_ctrl; // Data Offset (4 bits), Reserved (6 bits), Control bits (6 bits)

USHORT window; // (16 bits)

USHORT checksum; // (16 bits)

USHORT urgent\_pointer; // (16 bits)

} TCPHeader;

typedef struct UDPHeader

{

USHORT source\_port;

USHORT destination\_port;

USHORT length;

USHORT checksum;

} UDPHeader;

struct DNSHeader

{

USHORT id;

#if BYTE\_ORDER == LIL\_ENDIAN

USHORT recursion\_desired : 1;

USHORT truncated\_message : 1;

USHORT authoritive\_answer : 1;

USHORT operation\_code : 4;

USHORT is\_response : 1;

USHORT response\_code : 4;

USHORT checking\_disabled : 1;

USHORT authenticated\_data : 1;

USHORT reserved : 1;

USHORT recursion\_available : 1;

#elif BYTE\_ORDER == BIG\_ENDIAN

USHORT is\_response : 1;

USHORT operation\_code : 4;

USHORT authoritive\_answer : 1;

USHORT truncated\_message : 1;

USHORT recursion\_desired : 1;

USHORT recursion\_available : 1;

USHORT reserved : 1;

USHORT authenticated\_data : 1;

USHORT checking\_disabled : 1;

USHORT response\_code : 4;

#else

# error BYTE\_ORDER not defined.

#endif

USHORT question\_count;

USHORT answer\_count;

USHORT authority\_record\_count;

USHORT additional\_record\_count;

};

IPHeader\* hdr = (IPHeader \*)Buffer;

TCPHeader\* tcp\_hdr = (TCPHeader\*)(Buffer + IP\_HDR\_LEN);

UDPHeader\* udp\_hdr = (UDPHeader\*)(Buffer + IP\_HDR\_LEN);

DNSHeader\* dns\_hdr;

#define DNS\_QUERY\_DATA\_LEN 4

struct DNSQuestionData

{

unsigned short question\_type;

unsigned short question\_class;

};

struct DNSQuestion

{

unsigned char\* name;

DNSQuestionData\* data;

};

#define DNS\_RECORD\_DATA\_LEN 10

struct DNSRecordData

{

unsigned short record\_type;

unsigned short record\_class;

unsigned int ttl;

unsigned short response\_length;

};

struct DNSRecord

{

unsigned char\* name;

DNSRecordData\* data;

unsigned char\* response;

};

char \* ConvertToBinary(int b)

{

char s[10];

int c = b;

for (int i = 0; i<8; i++)

{

s[7 - i] = (c % 2) + '0';

c = c / 2;

}

s[8] = '\0';

return s;

}

unsigned short ip\_sum\_calc(char \*buffer, int len)

{

unsigned short word16;

char myBuf[80];

memcpy(myBuf, buffer, len);

myBuf[10] = '\0';

myBuf[11] = '\0';

unsigned sum = 0;

int i;

// разбиваем заголовок на 16-битные слова и суммируем их

for (i = 0; i<len; i = i + 2) {

word16 = ((myBuf[i] << 8) & 0xFF00) + (myBuf[i + 1] & 0xFF);

sum = sum + (unsigned long)word16;

}

// из 32-битной суммы получаем 16-битное слово, складывая старшее слово с младшим

while (sum >> 16)

sum = (sum & 0xFFFF) + (sum >> 16);

// побитовое дополнение

sum = ~sum;

return ((unsigned short)sum);

}

int parse\_dns\_query(unsigned char\* buffer, int offset)

//unsigned char\* parse\_dns\_query(unsigned char\* buffer, int offset)

{

char buf[1024] = { 0 };

char tmp[100] = { 0 };

offset += 12;

unsigned char \*curr = buffer + offset;

while (\*curr) {

strncpy(tmp, (const char\*)curr + 1, \*curr);

strcat(tmp, ".");

strcat(buf, tmp);

memset(tmp, '\0', sizeof(tmp));

offset += \*curr + 1;

curr += \*curr + 1;

}

buf[strlen(buf) - 1] = '\0';

printf("\t\tDNS lookup:\n\t\t target=%s\n", buf);

unsigned short\* query\_type = (unsigned short\*) (curr+3);

unsigned short\* query\_class = (unsigned short\*) (curr+1);

printf("\t\t query type=%u\n", htons(\*query\_type));

printf("\t\t query class=%u\n", htons(\*query\_class));

//return curr + 5;

return offset + 5;

}

unsigned short parse\_dns\_answer(unsigned char\* buffer, int offset)

{

char buf[1024] = { 0 };

char tmp[100] = { 0 };

unsigned char \*curr = buffer + offset;

unsigned char LLLL = \*(curr);

unsigned long long l1 = \*(curr);

if (LLLL & 0xC0) {

unsigned short off = htons((unsigned short)(\*curr)) & 0x3fff;

curr = buffer + off + 12;

}

while (\*curr) {

strncpy(tmp, (const char\*)curr + 1, \*curr);

strcat(tmp, ".");

strcat(buf, tmp);

memset(tmp, '\0', sizeof(tmp));

//offset += \*curr + 1;

curr += \*curr + 1;

}

if (LLLL & 0xC0) {

curr = buffer + offset + 1;

}

buf[strlen(buf) - 1] = '\0';

printf("\t\tDNS answer:\n\t\t target=%s\n", buf);

unsigned short\* query\_type = (unsigned short\*)(curr + 3);

unsigned short\* query\_class = (unsigned short\*)(curr + 1);

unsigned long\* ttl = (unsigned long\*)(curr + 5);

unsigned short\* data\_length = (unsigned short\*)(curr + 9);

printf("\t\t query type=%u\n", htons(\*query\_type));

printf("\t\t query class=%u\n", htons(\*query\_class));

printf("\t\t data length=%u\n", htons(\*data\_length));

printf("\t\t addr=");

char dl;

int L = (htons(\*data\_length));

if (L == 4)

dl = '.';

else

dl = ':';

for (int i = 0; i < L; i++) {

unsigned char\* B = (curr + 11 + i);

if (dl == '.')

printf("%u", \*B);

else

printf("%x", \*B);

if ((L - i) > 1)

printf("%c", dl);

}

return offset;

}

int main()

{

WSADATA wsadata; // Инициализация WinSock.

SOCKET s; // Cлущающий сокет.

char name[128]; // Имя хоста (компьютера).

HOSTENT\* phe; // Информация о хосте.

SOCKADDR\_IN sa; // Адрес хоста

IN\_ADDR sa1; //

unsigned long flag = 1; // Флаг PROMISC Вкл/выкл.

//

fd\_set s\_set;

struct timeval wait\_;

int sel\_res;

char temp[128];

// инициализация

WSAStartup(MAKEWORD(2, 2), &wsadata);

s = socket(AF\_INET, SOCK\_RAW, IPPROTO\_IP);

gethostname(name, sizeof(name));

phe = gethostbyname(name);

ZeroMemory(&sa, sizeof(sa));

sa.sin\_family = AF\_INET;

sa.sin\_addr.s\_addr = ((struct in\_addr \*)phe->h\_addr\_list[0])->s\_addr;

bind(s, (SOCKADDR \*)&sa, sizeof(SOCKADDR));

// Включение promiscuous mode.

ioctlsocket(s, SIO\_RCVALL, &flag);

// Включение promiscuous mode.

if (ioctlsocket(s, SIO\_RCVALL, &flag) != 0)

{

printf("Для работы программы необходимы права администратора!\n");

return -1;

}

FD\_ZERO(&s\_set);

FD\_SET(s, &s\_set);

//

wait\_.tv\_sec = 0;

wait\_.tv\_usec = 500;

// Бесконечный цикл приёма IP-пакетов.

while (!kbhit())

{

FD\_ZERO(&s\_set);

FD\_SET(s, &s\_set);

sel\_res = select(0, &s\_set, 0, 0, &wait\_);

if (sel\_res <= 0) continue;

int count;

char str[10];

count = recv(s, Buffer, sizeof(Buffer), 0);

// обработка IP-пакета

if (count >= sizeof(IPHeader))

{

//IPHeader\* hdr = (IPHeader \*)Buffer;

//TCPHeader\* tcp\_hdr = (TCPHeader\*)(Buffer + IP\_HDR\_LEN);

//UDPHeader\* udp\_hdr = (UDPHeader\*)(Buffer + IP\_HDR\_LEN);

// cast dns header

//DNSHeader\* dns\_hdr;

// Проверяем, является ли пакет DNS

switch (hdr->iph\_protocol)

{

case IPPROTO\_TCP:

if (!(htons(tcp\_hdr->destination\_port) == 53 || htons(tcp\_hdr->source\_port) == 53))

continue;

dns\_hdr = (DNSHeader\*)(Buffer + IP\_HDR\_LEN + TCP\_HDR\_LEN);

break;

case IPPROTO\_UDP:

if (!(htons(udp\_hdr->destination\_port) == 53 || htons(udp\_hdr->source\_port) == 53))

continue;

dns\_hdr = (DNSHeader\*)(Buffer + IP\_HDR\_LEN + UDP\_HDR\_LEN);

break;

}

//Начинаем разбор пакета...

//Разбираем заголовок

printf("----------==========IP-Packet==========----------\r\nHeader:\r\n");

//Выводим версию IP-протокола

printf("Version: %i\r\n", ((hdr->iph\_verlen & 0xF0) >> 4));

//Выводим длину заголовка

printf("Header length: %i bytes\r\n", (hdr->iph\_verlen & 0xF) \* 4);

//Выводим байт обслуживания

strcpy(str, ConvertToBinary(hdr->iph\_tos));

printf("Differentiated Services Field: 0x%sb\r\n", str);

// Вычисляем и выводим размер пакета. Так как в сети принят прямой порядок

// байтов, а не обратный, то прийдётся поменять байты местами.

printf("Total length: %u\r\n", htons(hdr->iph\_length));

//Выводим номер последовательности

printf("Identification: %u\r\n", htons(hdr->iph\_id));

//Выводим флаги

strcpy(str, ConvertToBinary(hdr->iph\_offset));

str[3] = '\0';

printf("Flags: 0x%sb", str);

if (str[2] == '1' || str[1] == '1') {

strcpy(temp, " (");

if (str[1] == '1') {

strcat(temp, "Don't Fragment");

if (str[2] == '1') {

strcat(temp, ", ");

}

}

if (str[2] == '1') {

strcat(temp, "More Fragments");

}

strcat(temp, ")");

printf("%s", temp);

}

printf("\r\n");

//Выводим смещение в потоке данных

printf("Fragment offset: %u\r\n", htons(hdr->iph\_offset) & 0x1FFF);

//Выводим время жизни пакета

printf("Time to live: %u\r\n", hdr->iph\_ttl);

// Выводим протокол следующего уровня. Полный список этих констант

// содержится в файле winsock2.h

printf("Protocol: ");

switch (hdr->iph\_protocol)

{

case IPPROTO\_TCP:

printf("TCP");

break;

case IPPROTO\_UDP:

printf("UDP");

break;

//case IPPROTO\_ICMP:

// printf("ICMP");

// break;

//case 47:

// printf("GRE");

// break;

//default:

// printf("OTHER %i", hdr->iph\_protocol);

// break;

}

printf("\r\n");

printf("Header checksum: %x", hdr->iph\_xsum);

unsigned short correctCRC = ip\_sum\_calc(Buffer, (hdr->iph\_verlen & 0xF) \* 4);

if (hdr->iph\_xsum == htons(correctCRC)) printf("[correct]\r\n"); else printf("[not corrent]\r\n");

//Преобразуем в понятный вид адрес отправителя.

printf("Source: ");

sa1.s\_addr = hdr->iph\_src;

printf(inet\_ntoa(sa1));

printf("\r\n");

// Преобразуем в понятный вид адрес получателя.

printf("Destination: ");

sa1.s\_addr = hdr->iph\_dest;

printf(inet\_ntoa(sa1));

printf("\r\n");

// Печатаем заголовок TCP или UDP пакета...

switch (hdr->iph\_protocol)

{

case IPPROTO\_TCP:

printf("\t> TCP-Packet:\r\n");

printf("\tTCP source port: %d, ", htons(tcp\_hdr->source\_port));

printf("\tTCP destination port: %d\n", htons(tcp\_hdr->destination\_port));

printf("\tSequence Number: %d\n", htonl(tcp\_hdr->seq\_number));

printf("\tAcknowledgement Number: %d\n", htonl(tcp\_hdr->ack\_number));

printf("\tChecksumm: 0x%04x\n", htons(tcp\_hdr->checksum));

break;

case IPPROTO\_UDP:

printf("\t> UDP-Packet:\r\n");

printf("\tUDP source port: %d, ", htons(udp\_hdr->source\_port));

printf("\tUDP destination port: %d\n", htons(udp\_hdr->destination\_port));

printf("\tLength: %d\n", htons(udp\_hdr->length));

printf("\tChecksumm: 0x%04x\n", htons(udp\_hdr->checksum));

break;

}

// Печатаем заголовок DNS пакета...

printf("\t\t> DNS-Packet:\r\n");

printf("\t\tTransaction ID: 0x%04x \r\n", htons(dns\_hdr->id));

printf("\t\tFlags:\r\n");

printf("\t\t Is response: %d\n", dns\_hdr->is\_response);

printf("\t\t Opcode: %d\n", dns\_hdr->operation\_code);

printf("\t\t Truncated: %d\n", dns\_hdr->truncated\_message);

printf("\t\t Recursion desired: %d\n", dns\_hdr->recursion\_desired);

printf("\t\t Reserver: %d\n", dns\_hdr->reserved);

//printf(" Truncated: %d\n", dns\_hdr->authenticated\_data);

printf("\t\tQuestions: %d\n", htons(dns\_hdr->question\_count));

printf("\t\tAnswer RRs: %d\n", htons(dns\_hdr->answer\_count));

printf("\t\tAuthority RRs: %d\n", htons(dns\_hdr->authority\_record\_count));

printf("\t\tAdditional RRs: %d\n", htons(dns\_hdr->additional\_record\_count));

//unsigned char\* rr = parse\_dns\_query((unsigned char\*)dns\_hdr, 0);

int rr = parse\_dns\_query((unsigned char\*)dns\_hdr, 0);

if (htons(dns\_hdr->answer\_count))

parse\_dns\_answer((unsigned char\*)dns\_hdr, rr);

printf("\n\n");

}

}

closesocket(s);

WSACleanup();

return 0;

}

# Заключение

В процессе выполнения курсовой работы в соответствии с заданием был проанализирован теоретический материал по сетевому стеку TCP/IP и протоколу DNS, разработано приложение для захвата и анализа пакетов протокола DNS.

# Список литературы

1. Керниган, Б. Язык программирования С: учеб.пособие/ Б.Керниган, Д.Ритчи-2-е изд. 2009.-304 с.
2. Ворожцов, A.B. Лекции. Алгоритмы: построение, анализ и реализация на языке программирования Си: учеб.пособие / Ворожцов A.B., Винокуров Н.А.-МФТИ,2007 - 452 с.
3. Подбельский, В. В. Программирование на языке Си: учеб.пособие/ Подбельский, С. С. Фомин-финансы и статистика,2005 – 600 с.
4. Гулаков B.K. Многомерные структуры данных: учеб.пособие / B.K. Гулаков, A.O. Трубаков - Брянск: БГТУ, 2010. - 387 с.
5. Кнут Д., Искусство программирования, т.З. M.: Вильямс, 2000.
6. Левитин, A.B. Алгоритмы: введение в разработку и анализ: Учеб, пособие/ A.B. Левитин. - М.: Изд-во Вильямс, 2006. - 576 с.
7. Кормен, X.T. Алгоритмы: построение и анализ: Учеб, пособие/ X.T. Кармен, Ч.Э. Лейзерсон, Р.Л. Ривест, К. Штайн. - МлВильямс, 2006 – 536 с.