Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Уральский федеральный университет имени первого Президента России Б.Н. Ельцина» (УрФУ)

Институт радиоэлектроники и информационных технологий – РтФ

Школа профессионального и академического образования

Отчет по дисциплине   
«Безопасность компьютерных сетей»

Лабораторная работа №3  
«Знакомство с сетевым анализатором Wireshark»

Студенты: Клоченко И.Е., Юсупов Д.А.

Преподаватель: Ваулин С.С.

Группа: РИ-300024

Екатеринбург

2023

**Оглавление**

[1. Знакомство с интерфейсом приложения 3](#_Toc132967636)

[2. Фильтрация пакетов 5](#_Toc132967637)

[3. Анализ служебных протоколов 8](#_Toc132967638)

[4. Обращение к *web*-сайту 15](#_Toc132967639)

[Вывод 20](#_Toc132967640)

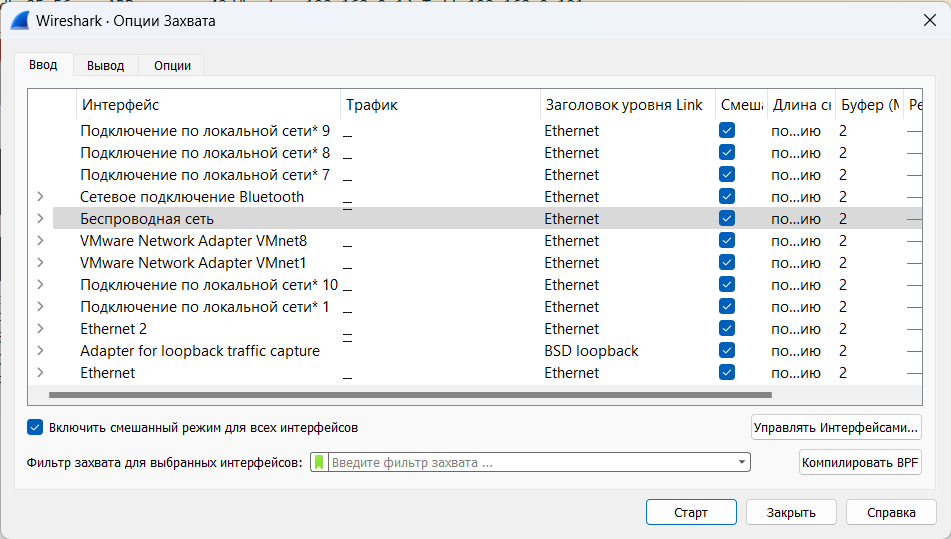
**Цель работы:**

Знакомство с сетевым анализатором трафика *Wireshark*, изучение приемов анализа трафика.

Ход работы:

## Знакомство с интерфейсом приложения

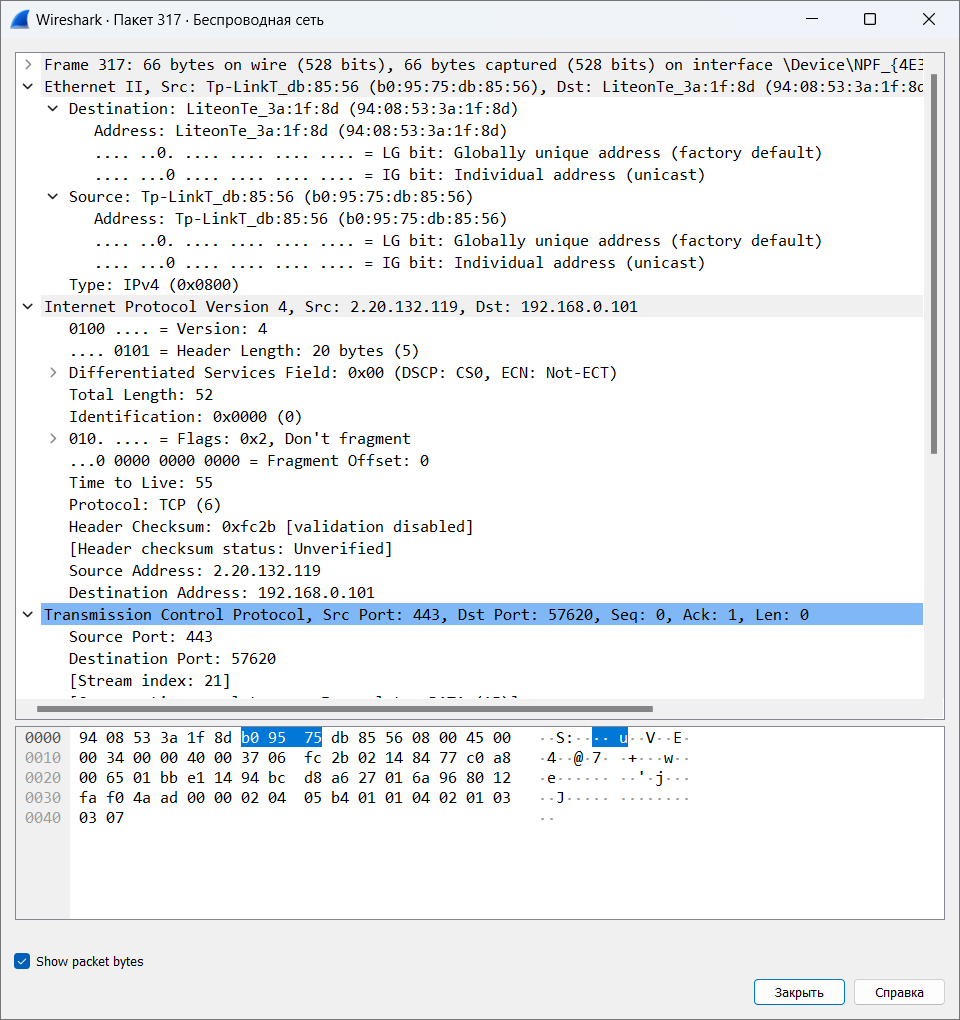
* 1. Запускаем *Wireshark*. Перечислим основные доступные для захвата трафика интерфейсы:



**Рисунок 1.** Основные интерфейсы для анализа и захвата трафика в Wireshark

С помощью *ipconfig* и *route* определим, какой из интерфейсов используется для отправки пакетов шлюзу по умолчанию – это Беспроводной адаптер *Wi*-*Fi*.

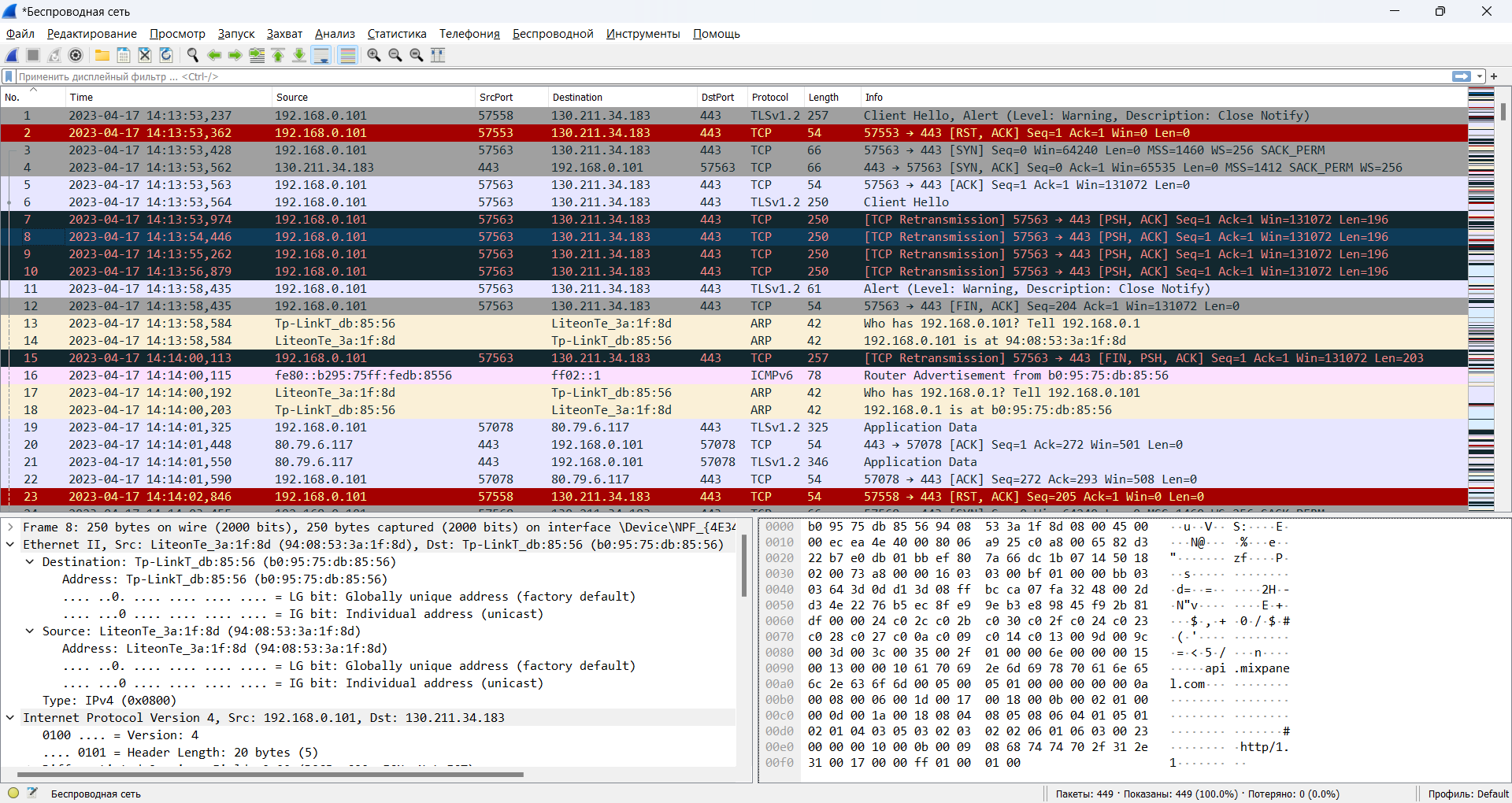
* 1. Продолжаем захват трафика в течение минуты, затем останавливаем его. Просматриваем захваченные пакеты. Прикладные протоколы: только *DNS* и *HTTP*.
  2. Выбираем любой пакет и просматриваем вложенные протоколы:

****

**Рисунок 2. Вложенные протоколы в пакете**

Пакет отправило приложение *Akamai* *Technologies* **(2**.20.132.119). Пакет с флагами *SYN*, *ACK*, то есть «второе рукопожатие». Скорее всего этот пакет был отправлен этим сервером для подтверждения подключения через протокол *TCP* для подключения к какому-либо веб-ресурсу в интернете.

* 1. В списке захваченных пакетов добавим столбцы, содержащие порт источника и порт назначения. Изменим формат даты.

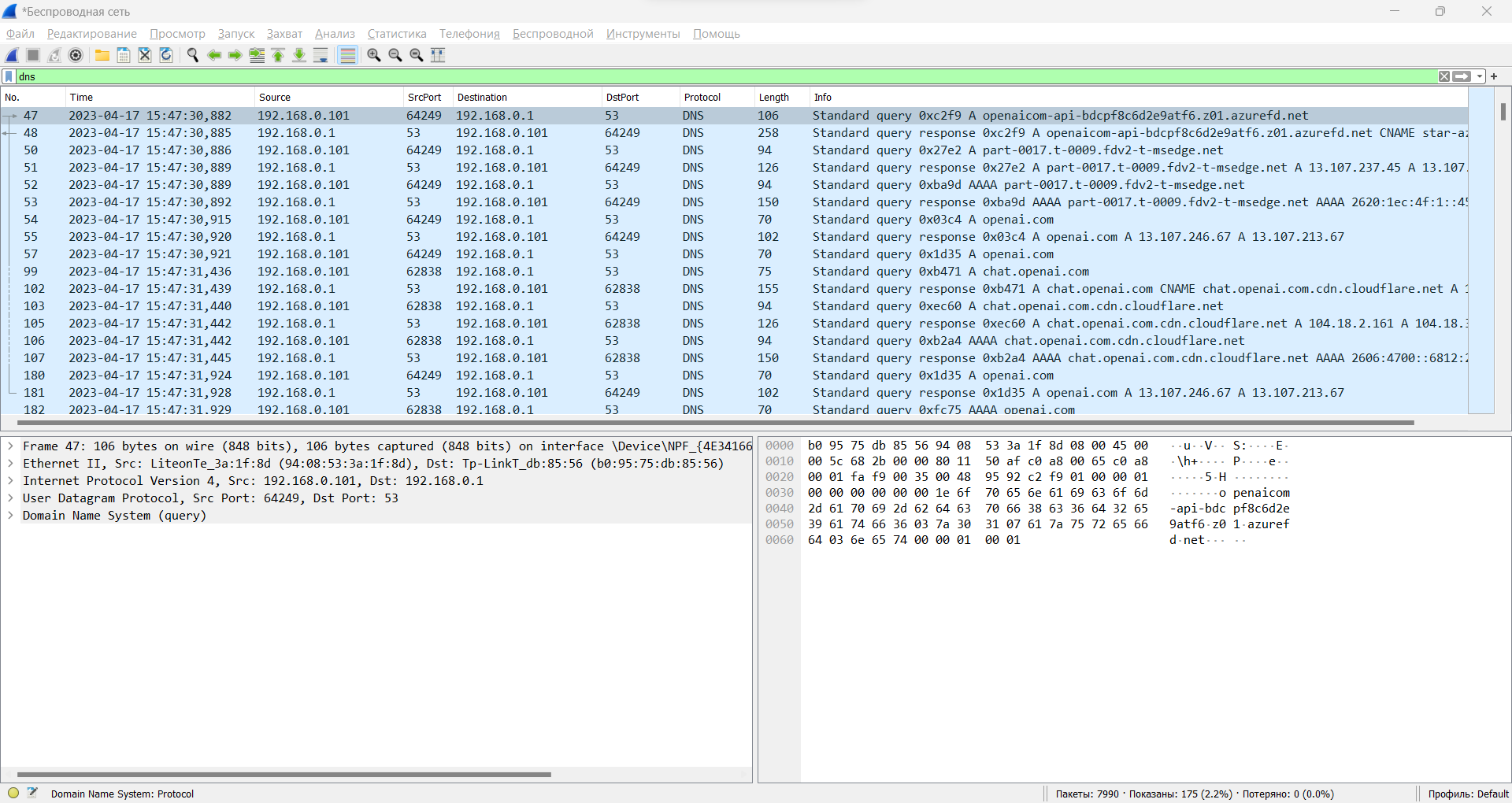


**Рисунок 3.** Настроенное отображение в 1 окне Wireshark

* 1. Сохраним полученный трафик в файл. Расширение: .*pcapng*. Размер: 94,4 КБ.

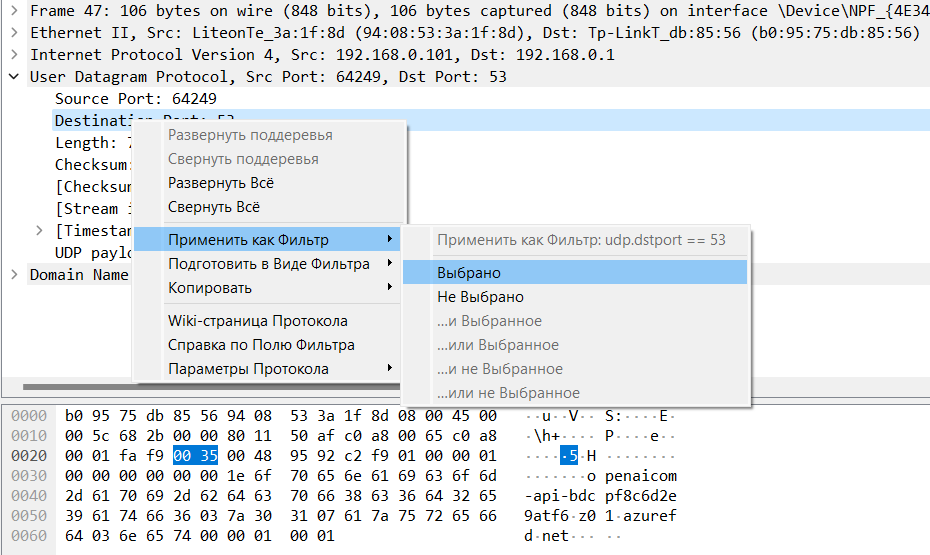
## Фильтрация пакетов

* 1. Запускаем захват трафика в *Wireshark* и переходим в веб-браузер на сайт *gmail*.*com*. Останавливаем захват пакетов. При помощи декодера пакетов создаем фильтр для отображения только *DNS* пакетов.



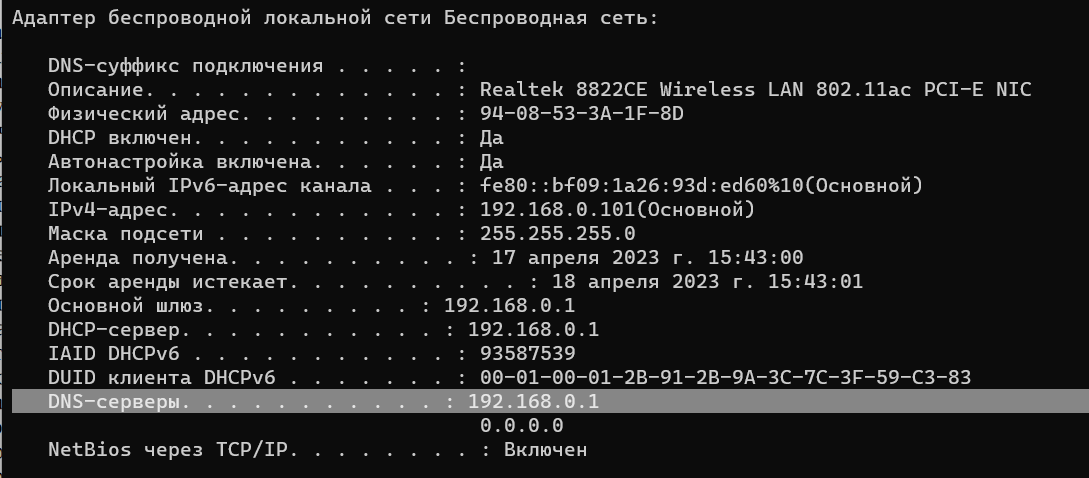
**Рисунок 4.** Фильтр «только DNS»

* 1. Очищаем фильтр и снова выбираем любой *DNS*-пакет. В заголовке *UDP* пакета находим порт 53 и используем это значение в качестве нового фильтра.



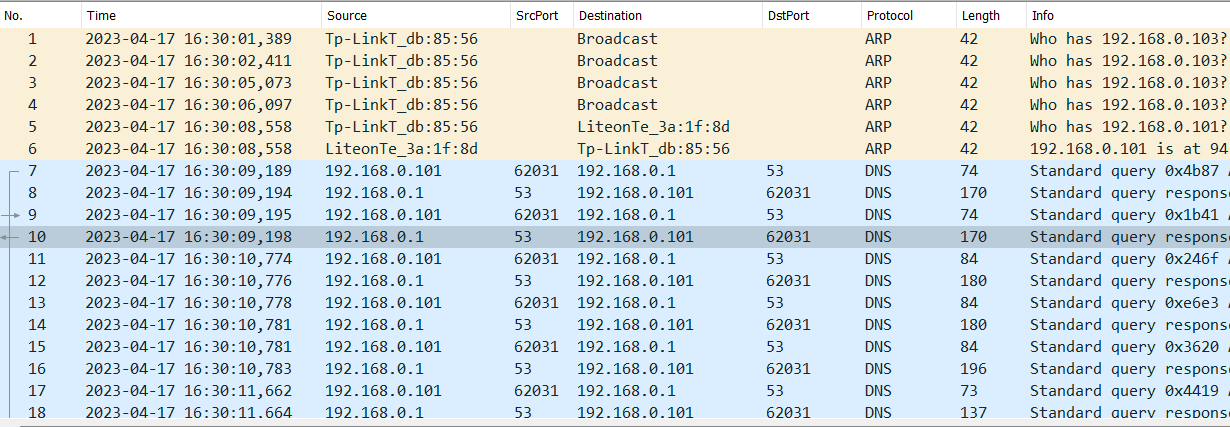
**Рисунок 5.** Использование поля с портом 53 в качестве нового фильтра

* 1. Отличие заключается в том, что в п. 2.2. в качестве нового фильтра был выбран *Destination* *port* (53). После его применения были отфильтрованы все пакеты, где *destination* *port* равен 53. В п. 2.1. отфильтрованы все пакеты, где используется порт 53 – *source* *port* или *destination* *port*.
  2. Выполняем команду *ipconfig* /*all* и в разделе основного сетевого адаптера находим строку с началом «*DNS*-сервер».



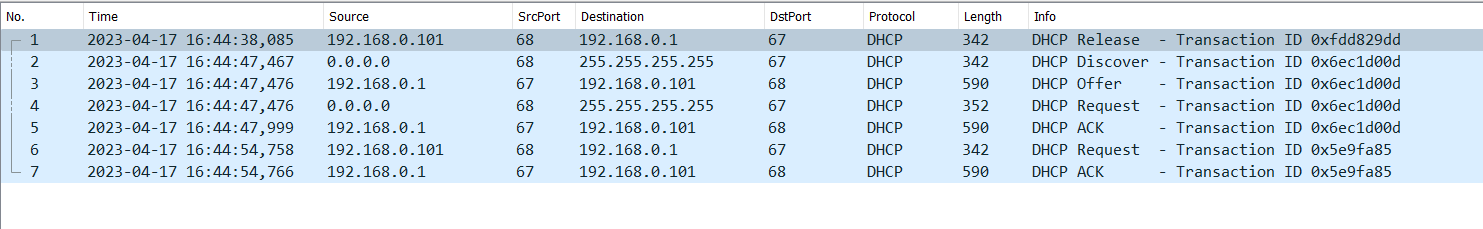
**Рисунок 6.** Узлы-адресанты DNS пакетов

* 1. Создадим фильтр захвата, который будет пропускать только *ARP* и *DNS* пакеты. Начнем захват трафика и продолжим его до тех пор, пока не будут захвачены пакеты двух искомых типов. Фильтр: *port* 53 *or* *arp*.



**Рисунок 7.** Действие фильтра захвата

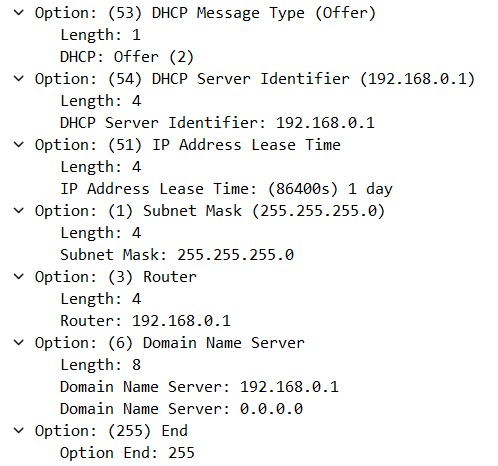
* 1. Создадим фильтр, используя обращение к байтам по их смещению относительно начала *Ethernet* кадра. Фильтр: *ether[*12:2*]*=0*x*0806.
  2. Запускаем захват трафика, используя фильтр захвата, пропускающий только *DHCP* пакеты. Фильтр: *port* 67 *or* *port* 68.
  3. Выполняем команды *ipconfig* /*release* и *ipconfig* /*renew*. Останавливаем захват трафика.



**Рисунок 8.** Захваченные DHCP пакеты

Типы *DHCP* пакетов:

* *Release* – освобождение *IP* адреса.
* *Discover* – первый пакет, который отправляется клиентом при поиске *DHCP*-сервера для получения *IP* адреса.
* *Offer* – ответ *DHCP*-сервера на *DHCP* *Discover* с сообщением о *IP* адресе, который может использоваться.
* *Request* – ответ клиента на *DHCP* *Offer*, чтобы запросить *IP* адрес (может отсылаться для продления аренды уже используемого *IP* адреса).
* *ACK* – ответ *DHCP* сервера на *DHCP* *Request*, который содержит *IP* адрес, который можно использовать, длительность его аренды и информацию о адресах *DNS* серверов и шлюз по умолчанию.
  1. Пакет *DHCP* *Offer* предоставляет следующие сетевые настройки для клиента:

****

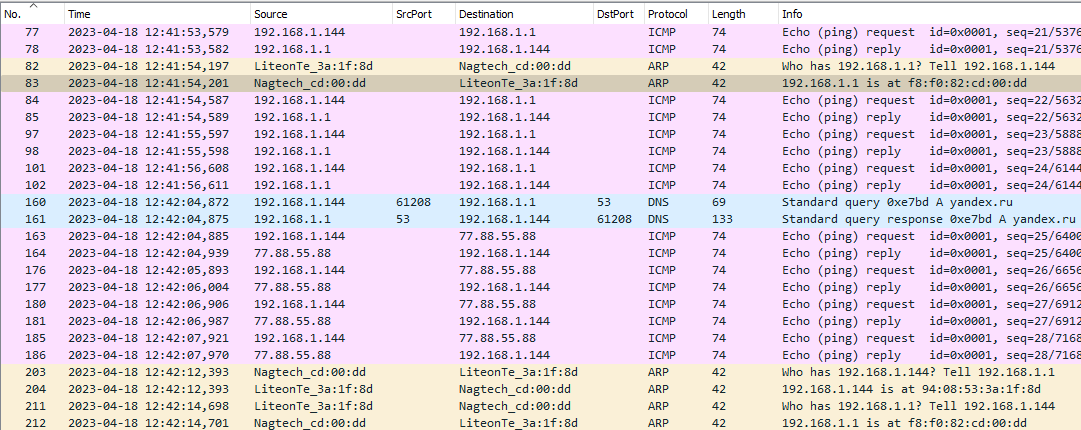
**Рисунок 9.** Настройки, предоставляемые DHCP Offer

Среди них:

* Адрес *DHCP* сервера
* Маска сети
* Шлюз по умолчанию
* Адреса *DNS* сервера

## Анализ служебных протоколов

* 1. Запускаем *Wireshark* и начинаем захват трафика. Запускаем командную строку от имени администратора и очищаем *ARP*-таблицу. Выполняем эхо-тестирование сначала шлюза по умолчанию, а затем узла *yandex*.*ru*. Завершаем захват трафика. Скрываем лишний трафик с помощью фильтров отображения (только *ARP*, *DNS*, *ICMP* пакеты).

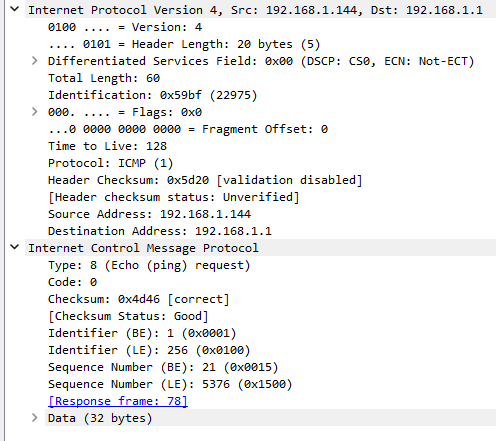
****

**Рисунок 10**. Захваченные пакеты после очищения arp-таблицы и эхо-тестирования шлюза по умолчанию и узла yandex.ru

* 1. Во время захвата пакетов было захвачено 6 *ARP* пакетов 2 типов: *ARP* *Request* и *ARP* *Reply*. Узлы отправители/получатели: адаптер беспроводной локальной сети и шлюз по умолчанию. *ARP* пакеты появились в результате эхо-тестирования, так как мы предварительно очистили *arp*-таблицу (то есть записей о шлюзе по умолчанию там не стало). Выполнив эхо-тестирование, мы снова «познакомили» ПК с ним, внеся записи в *arp*-таблицу.
  2. Во время захвата пакетов было захвачено 2 *DNS* пакета типов: *Query* и *Response*. Узлы отправители/получатели на сетевом и канальном уровнях – это *DNS*-сервер (тот же *IPv*4 адрес, что и у шлюза по умолчанию) и локальный компьютер. Во втором пакете они просто меняются местами.

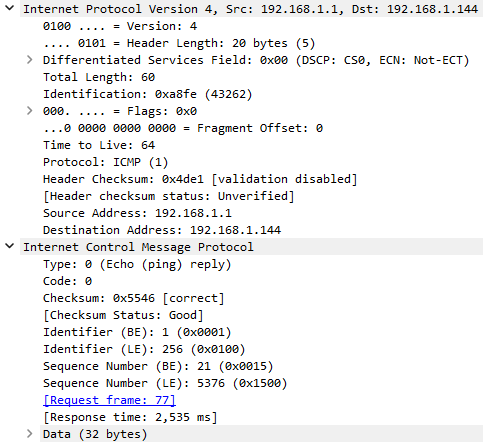
При выполнении команды (*ping* *yandex*.*ru*) идет обращение к *DNS* серверу, чтобы получить *IPv*4 адрес доменного имения *yandex*.*ru* для дальнейших обращений к узлу.

* 1. Среди захваченных *ICMP* пакетов выбираем один случайный. Фиксируем содержимое декодера для заголовков *IP* и *ICMP* выбранного пакета:



**Рисунок 11**. Содержимое декодера для заголовков IP и ICMP выбранного пакета

Зафиксируем на содержимое декодера пакетов заголовки *IP* и *ICMP* ответного пакета:



**Рисунок 12**. Содержимое декодера для заголовков IP и ICMP ответного пакета

Связь между *ICMP* *request* и *ICMP* *reply* пакетами задается полями *Identifier* (*BE*/*LE*), а если у двух пакетов одинаковый идентификатор, то еще полями *Sequence* *Number* (*BE*/*LE*).

* 1. Создадим фильтр, который позволит отобразить только *ICMP* пакеты, которые использовались для эхо тестирования узла *yandex*.*ru*. Для начала нужно взять *IPv*4 адрес из колонки *Destination*. Затем подставить этот адрес в выражение для фильтра отображения. Итоговый фильтр: *icmp* *and* *ip*.*addr* == 77.88.55.88.

Для всех захваченных *ICMP* пакетов при эхо-тестировании узла *yandex*.*ru*:

Отправитель: 192.168.1.144, 94-08-53-3*A*-1*F*-8*D*

Получатель: 77.88.55.88, *f*8:*f*0:82:*cd*:00:*dd*.

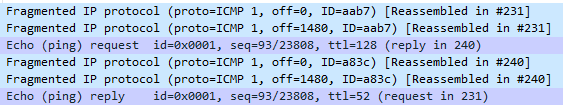
Один и тот же пакет на сетевом и канальном уровне может быть адресован разным узлам (*MAC* адрес *f*8:*f*0:82:*cd*:00:*dd* не соответствует узлу с *IP* адресом 77.88.55.88) по причине маршрутизации. Во время маршрутизации *MAC* адрес постоянно меняется, в отличии от логического адреса (*IP*). Прежде чем пакет достигнет конечного адреса, он проходит через несколько узлов, у которых свои *MAC* адреса. На канальном уровне отправитель-получатель – это его «соседи», а на сетевом – начальный и конечный *IP* адреса (в нашем случае *IP* адрес локального компьютера и *IP* адрес узла *yandex*.*ru*).

* 1. Выполним эхо тестирование узлов *yandex*.*ru* и *google*.*com* пакетами длиной 1400 байт с захватом трафика.

В поле данных для запросов и ответов у *ICMP* пакетов адресованных и *yandex*.*ru*, и *google*.*com*, содержится (кроме заголовков) латинские буквы *a*-*z*, которые заполняют все 1400 байт. Длины также одинаковы для обоих серий захваченных пакетов – 1400, за исключение пакетов ответа для эхо-тестирования узла *google*.*com*. В этом пакете длина пакета 68 байт.

* 1. Выполним эхо-тестирование узла *yandex*.*ru* пакетом длиной 3000 байт с захватом трафика. Используем фильтр отображения *icmp*. В списке захваченных пакетов отсутствуют пакеты ответов от узла (в это время в терминале появляются сообщения «Превышен интервал ожидания для запроса»). Это происходит по причине большого размера для передачи в одном кадре. Пакеты разбиваются на фрагменты и передаются по отдельности.

Используем фильтр отображения *ip*.*addr* == 77.88.55.60 и находим в списке «пропавшие пакеты». В поле *info* можно найти следующие строки:



**Рисунок 13**. Фрагментированные пакеты ICMP (1 пакет эхо-запроса с ответом)

Находим все три фрагмента одного любого эхо-запроса. Выпишем значения полей «*Identification*», «*MF*», «*Fragment* *offset*» *IP*-заголовка для этих пакетов.

Таблица 1. Фрагментированные части одного эхо-запроса

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Identification** | **MF (More fragment)** | **Fragment offset** |
| **1** | 43068 | *Set* | 0 |
| **2** | 43068 | *Set* | 1480 |
| **3** | 43068 | *Not* *set* | 2960 |

*Identification* позволяет определить относится ли пакет к части фрагментированного пакета.

*More* *fragment* (*MF*) позволяет определить, есть ли еще пакеты, которые являются частью фрагментированного пакета.

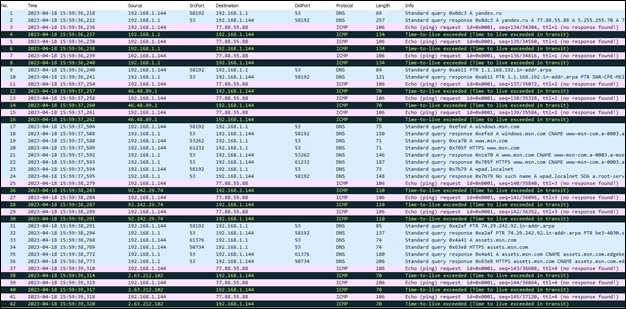
*Offset* *fragment* позволяет определить порядок следования фрагментированных пакетов. Следует выставлять порядок следования пакетов по возрастанию значения этого поля.

* 1. Выполним эхо-тестирование узла *yandex*.*ru* с ключом -*r* 9 с захватом трафика. Информация о маршруте содержится в захваченном пакете по следующему пути:

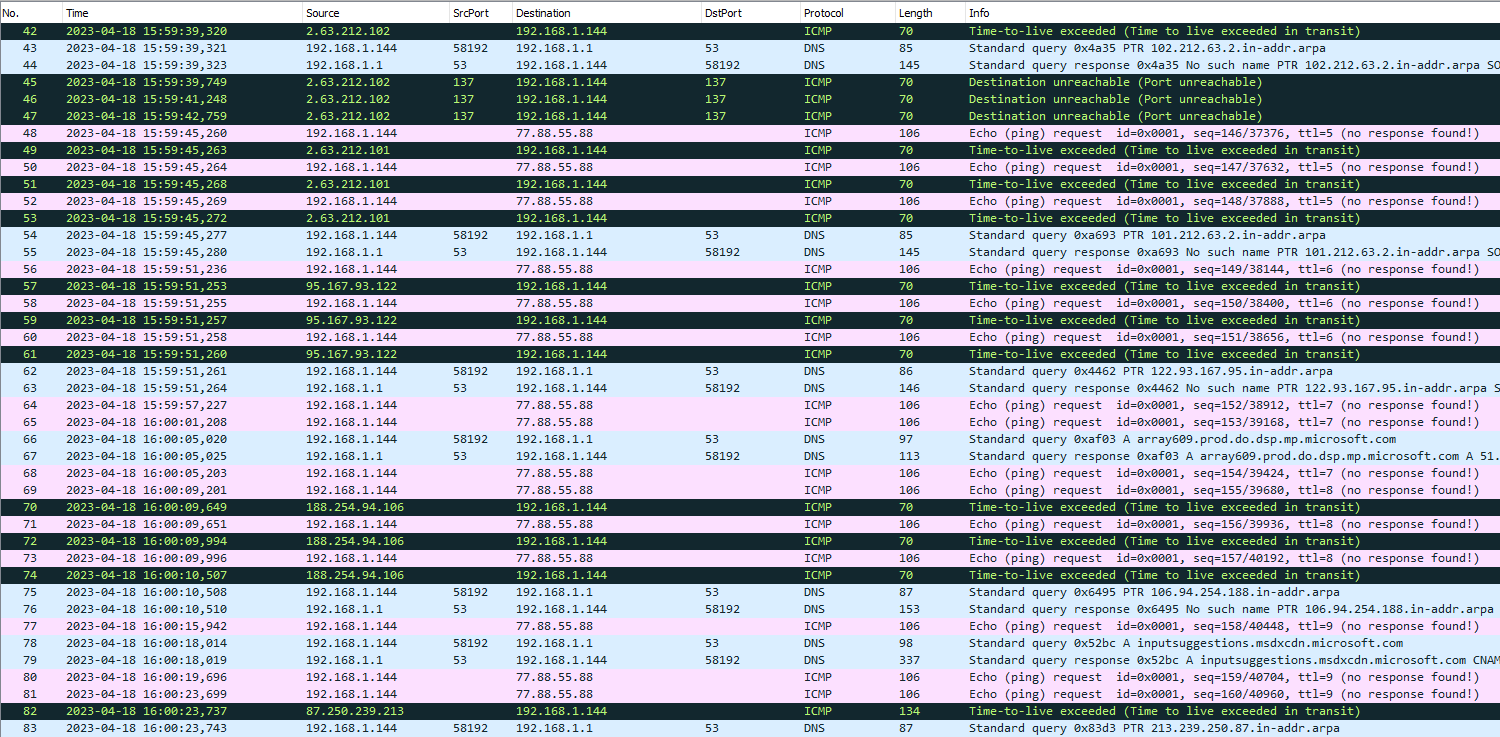
Заголовок *IPv*4 – *Option*: (40 *bytes*), *Record* *Route*.

9 максимально возможное число для параметра -*r* потому что если выставить, к примеру 10, то во первы, утилита выдаст ошибку «Нельзя так делать, значение должно быть с 1 до 9», а во вторых, так как параметр -*r* задает количество прыжков (или *TTL*) которое должно быть пройдено до целевого узла, то 9 – оптимальное для времени ответа от узла число, а еще лишающее возможности получить ошибку при передаче пакета на одном из очередных узлов.

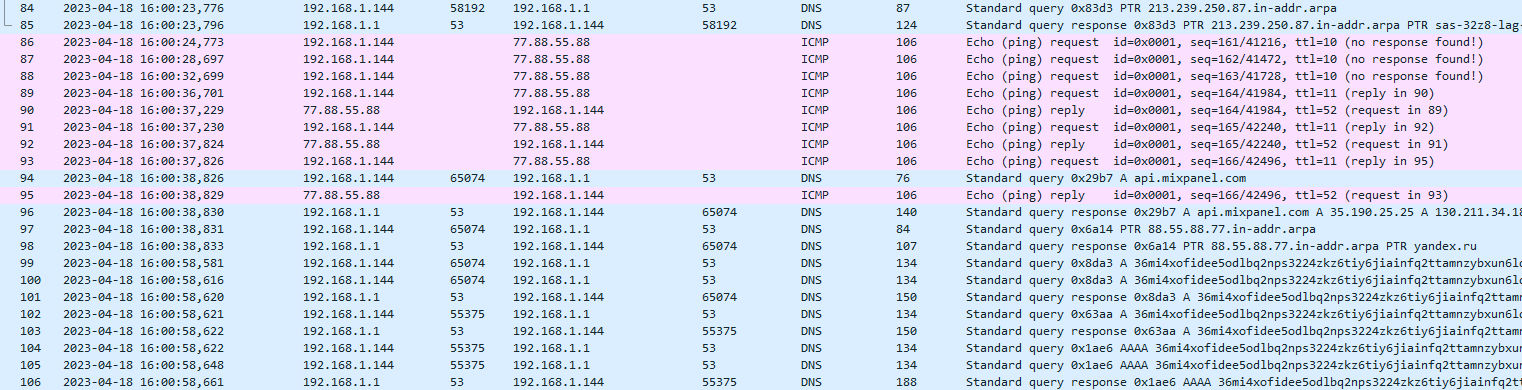
* 1. Создадим фильтр захвата трафика, который пропускает только *ICMP* и *DNS* пакеты: *icmp* *or* (*port* 53 *or* *port* 67). Начинаем захват трафика с созданным фильтром, а затем выполняем трассировку узла *yandex*.*ru*. Ждем выполнения трассировки и останавливаем захват трафика. Результат представлен в 3 снимках экрана (большой объем захваченных пакетов).



**Рисунок 14**. Захваченные пакеты при трассировке узла yandex.ru (1)



**Рисунок 15**. Захваченные пакеты при трассировке узла yandex.ru (2)



**Рисунок 16**. Захваченные пакеты при трассировке узла yandex.ru (3)

* 1. Были захвачены следующие типы *ICMP* сообщений:
* *Time* *to* *live* *exceeded* (*Time* *to* *live* *exceeded* *to* *transit*) – присылается очередным узлом на пути к целевому.
* *Destination* *unreachable* (*Port* *unreachable*) – присылается узлами, если интервал ожидания был превышен.
* *Echo* (*ping*) *request* (*reply*) – присылается всеми узлами на маршруте до целевого узла.

Различные узлы присылают сообщение с типом 11 в ответ запрос к одному и тому же узлу, потому что на очередной шаге *TTL* уменьшается, и, если *TTL* на узле равно 0, он уничтожает *ICMP* пакет и отправляет источнику (нашему локальному компьютеру), ответом с типом сообщения «*Time* *to* *live* *exceeded* (*Time* *to* *live* *exceeded* *to* *transit*)», чтобы тот в свою очередь уже знал маршрут до этого узла и в следующий раз отправлял пакет с *TTL*+1.

* 1. Отфильтруем пакеты, оставив только эхо-запросы.

Остались сообщения с типом сообщения 11 по следующей причине. В *RFC* 792 указан формат заголовка для *ICMP* пакета с типом сообщения 11: «Интернет-заголовок плюс первые 64 бита оригинала данные дейтаграммы. Эти данные используются хостом для сопоставления сообщение соответствующему процессу». В эти 64 бита оригинала пакета попадает ссылка на оригинальный пакет (пакет *ICMP* с типом сообщения 8), из-за чего пакет с типом сообщения 11 проходит через фильтр.

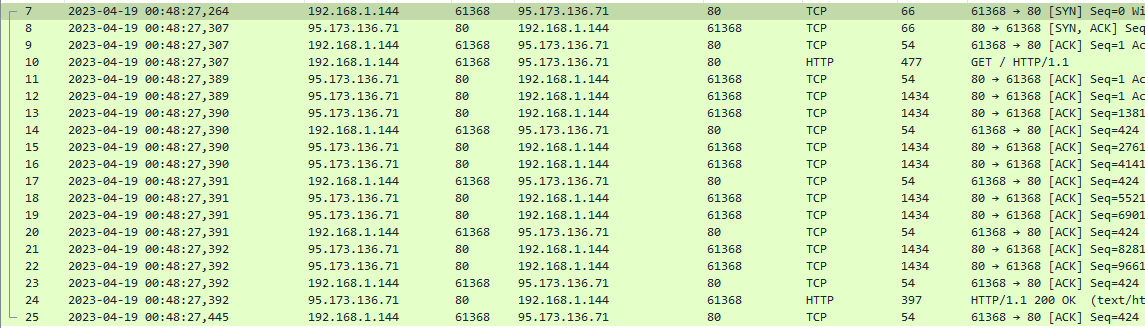
Пробуем модифицировать фильтр: *icmp*.*type* == 8 *and* *not* *icmp*.*type* == 11. И теперь мы видим в списке захваченных пакетов только исходные запросы.

* 1. В списке захваченных пакетов есть *DNS* пакеты. Их роль в процессе трассировки следующая. Когда мы обращаемся к узлу по доменному имени, мы обращаемся к *DNS* серверу, чтобы получить *IPv*4 адрес. Тип *DNS* записей – *A* (получение *IPv*4 адреса для запрашиваемого доменного имени *yandex*.*ru*).

## Обращение к *web*-сайту

* 1. Запустим захват трафика в *Wireshark* и откроем сайт [*http*://*linuxatemyram*.*ru*](http://linuxatemyram.ru) (сайт ни в какую не хочет загружаться через *http*, во всех браузерах загружается через *https*; возможно это связано с тем, что оригинальный сайт, который можно найти через поисковую систему, имеет *https* протокол).

Используем другой сайт для работы. Например, [*http*://*kremlin*.*ru*/](http://kremlin.ru/). Посмотрим, какой *IP* адрес был использован для запроса содержимого сайта и используем его для создания фильтра отображения.



**Рисунок 17**. Список пакетов при загрузке сайта

* 1. В списке захваченных пакетов найдем тот, у которого текст в столбце *Info* начинает с «*GET* / *HTTP*». Выполним «*Follow* → *TCP* *Stream*».



**Рисунок 18**. Содержимое TCP потока

* 1. Фильтр отображения принял значение «*tcp*.*stream* *eq* 1». У первых трех пакетов в потоке следующие флаги: *SYN*; *SYN*, *ACK*; *ACK*.

Опции «*MSS*» и «*Windows* *scale*» изменяются у первых двух пакетов с флагами *SYN*; *SYN*, *ACK*;

Опция *MSS* определяет максимально возможный размер сегмента данных для одного *TCP* пакета (по умолчанию 536 байт).

Опция *Window* *scale* – это множитель для увеличения размера окна передачи данных.

Определим, меняется ли размер окна в потоке. Смотрим на поле *[Calculated* *window* *size*: *NNNN]*. Размер окна меняется на протяжении всего потока.

* 1. Для всех запросов от клиента к серверу размер окна составляет 512 и не меняется (за исключение последнего пакета от клиента, там значение 510).

Для всех запросов от сервера к клиенту размер окна фиксирован (237).

Исключить из этого сравнения стоит первые 3 пакета, в них размеры окон резко отличаются от всех остальных.

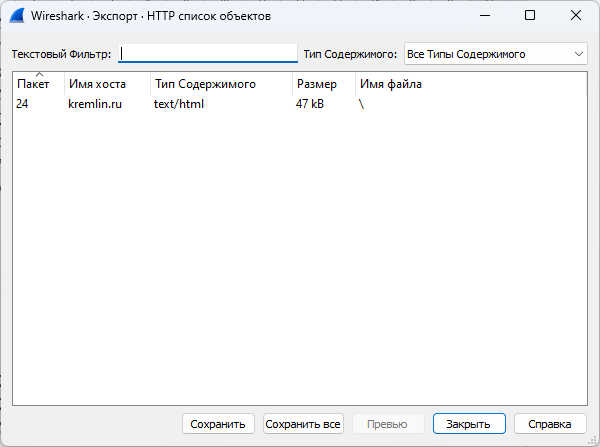
* 1. Выпишем все заголовки, использованные в *HTTP* запросе и ответе.

Запрос:

* *Host* – имя хоста
* *User*-*Agent* – идентификатор браузера или иного клиента
* *Accept* – форматы контента, которые может обрабатывать клиент
* *Accept*-*Language* – языки, которые может обрабатывать клиент
* *Accept*-*Encoding* – методы сжатия контента, которые может обрабатывать клиент
* *Connection* – тип соединения (в наше случае *keep*-*alive*)
* *Cookie* – передача данных о сессии между клиентом и сервером
* *Upgrade*-*Insecure*-*Requests* – указывает для клиентского браузера использовать защищенный протокол *HTTPS* (даже если ресурс использует протокол *HTTP*)
* *DNT* – «не отслеживать мою активность»

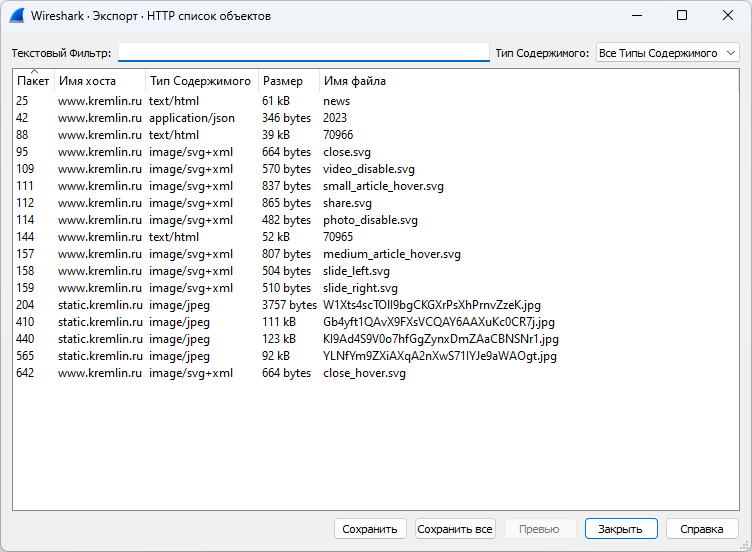
Ответ:

* *Server* – информация о сервере, обрабатывающем запрос
* *Date* – Дата и время составления запроса сервером
* *Content*-*Type*
* *Transfer*-*Encoding* – метод кодирования для передачи тела сообщения
* *Connection*
* *Keep*-*Alive* – указатель на сохранение соединения между клиентом и сервером после завершения текущего ответа
* *Vary* – указатель на то, какие заголовки запроса должны использоваться при кэшировании данного ответа
* *X*-*Frame*-*Options* – защита от кликджекинга.
* *X*-*Content*-*Type*-*Options* – защита от атак *MIME*.
* *Content*-*Security*-*Policy* – управление и ограничение загрузки контента на страницу, тем самым предотвращая атаки *XSS* и *SQLi*.
* *X*-*UA*-*Compatible* – указатель на то, какой режим отображения следует использовать браузеру при открытии страницы.
* *Content*-*Encoding* – указатель на способ сжатия тела *HTTP* сообщения.
  1. Очистим фильтры отображения и остановим захват пакетов. Перейдем «*File* → *Export* *Objects* → *HTTP*». В списке присутствуют следующие файлы:



**Рисунок 19**. Список файлов во встроенном декодере для прикладного уровня (1)

В списке присутствует одна строка. Пробуем заново, но теперь не просто загрузим страницу, а сделаем несколько переходов и полистаем фотографии на сайте (все файлы на этом сайте подгружаются отдельно от загрузки самой страницы).

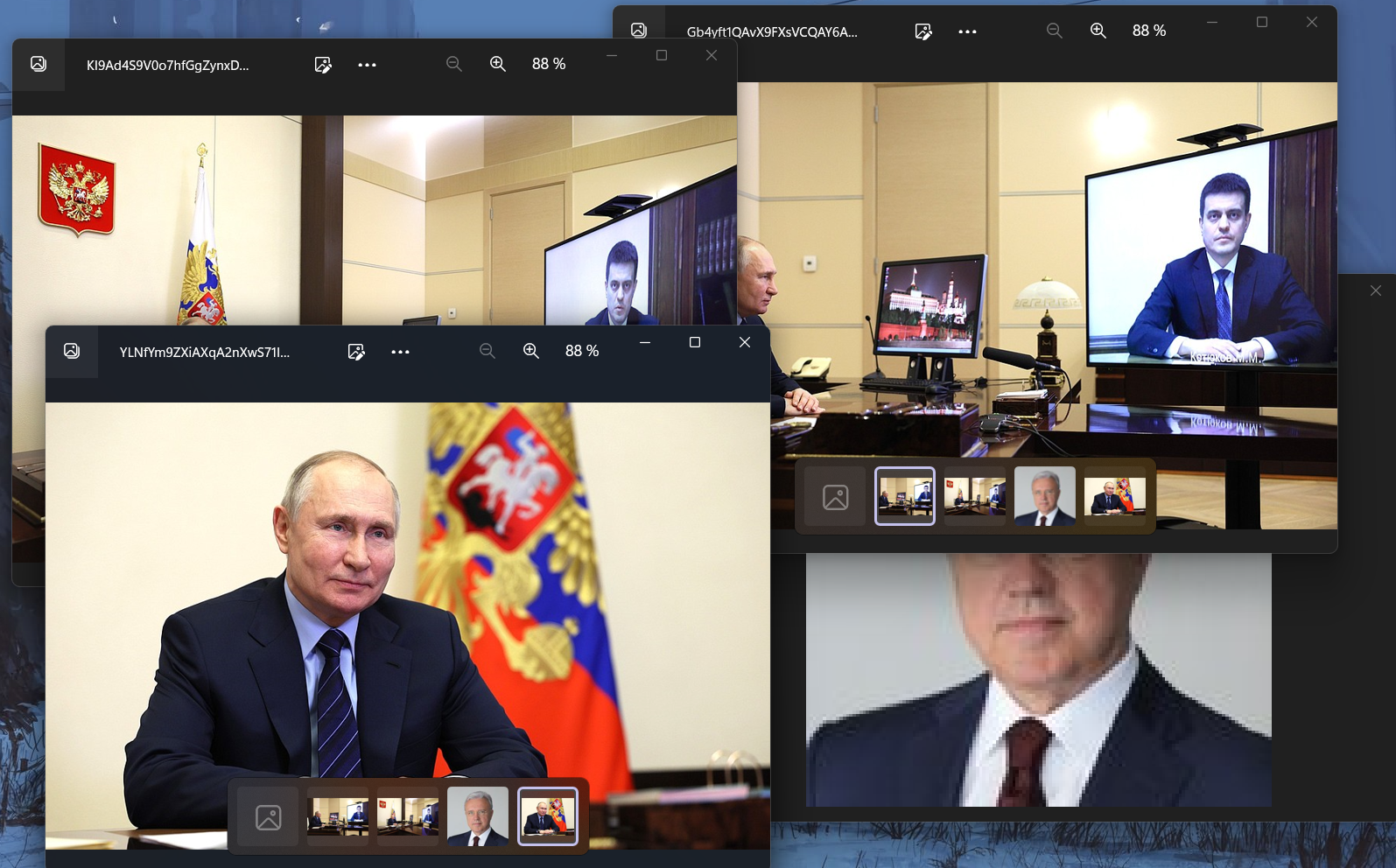


**Рисунок 20**. Список файлов во встроенном декодере для прикладного уровня (2)

В списке присутствуют такие файлы, как

* *text*/*html* – макет (разметка) сайта
* *application*/*json* – *MIME* тип для обмена скриптами *JSON* между клиентом и сервером
* *image*/*svg*+*xml* – масштабируемая векторная графика (язык разметки *XML*)
* *image*/*jpg* – изображение в формате *JPG*.

Так как изображений в формате *PNG* нет, обратимся к изображениям в формате *JPG*. Сохраним их на компьютер и откроем для просмотра.



**Рисунок 21**. Фотографии, экспортированные из HTTP пакетов

## Вывод

В ходе выполнения лабораторной работы был получен базовый опыт использования инструмента сетевого анализа *Wireshark*.

Изучены и практически применены способы фильтрации трафика: фильтр отображения и фильтр захвата трафика.

Изучены типы *ARP*, *DHCP*, *ICMP*, *DNS* пакетов с практическим отображением работы этих типов.

Для *ICMP* пакетов были рассмотрены поля, отвечающие за размер пакета при эхо-тестировании. Изучено содержимое *ICMP* пакета в разделе *Data*.

Также наглядно изучен принцип фрагментации пакетов со способами определить количество и порядок следования этих пакетов при эхо-тестировании.

Была проведена трассировка с параллельным захватом трафика. Ознакомились с типами *ICMP* сообщений.

Произведено обращение к *web*-сайту с параллельным захватом трафика. Проведен анализ пакетов *HTTP*. Изучены заголовки *HTTP*. Познакомились с такими опциями, как множитель размера окна передачи данных (*Window* *Scale*) и максимально возможный размер сегмента данных (*MSS*).

Выполнен захват пакетов во время загрузки веб-сайта и «серфинга» по нему. Освоено экспортирование объектов из *HTTP* пакетов.