ПРАКТИЧЕСКАЯ РАБОТА 2. АНАЛИЗ ПРОТОКОЛА HTTP

Цели работы

• Изучение основ работы протокола HTTP: понять основные принципы функционирования HTTP, его методы (GET, POST и др.) и структуру запросов и ответов.

• Овладение навыками работы с Wireshark: научиться использовать Wireshark для захвата и анализа сетевого трафика.

• Анализ HTTP-запросов и ответов: научиться идентифици

ровать и интерпретировать HTTP-запросы и ответы в захваченном трафике.

• Изучение кодов состояния HTTP: ознакомиться с различными кодами состояния ответов HTTP и их значениями.

• Исследование заголовков HTTP: понять роль заголовков в HTTP-запросах и ответах, изучить, как они влияют на поведение клиента и сервера.

Задание 1. Загрузка файла трассировки http

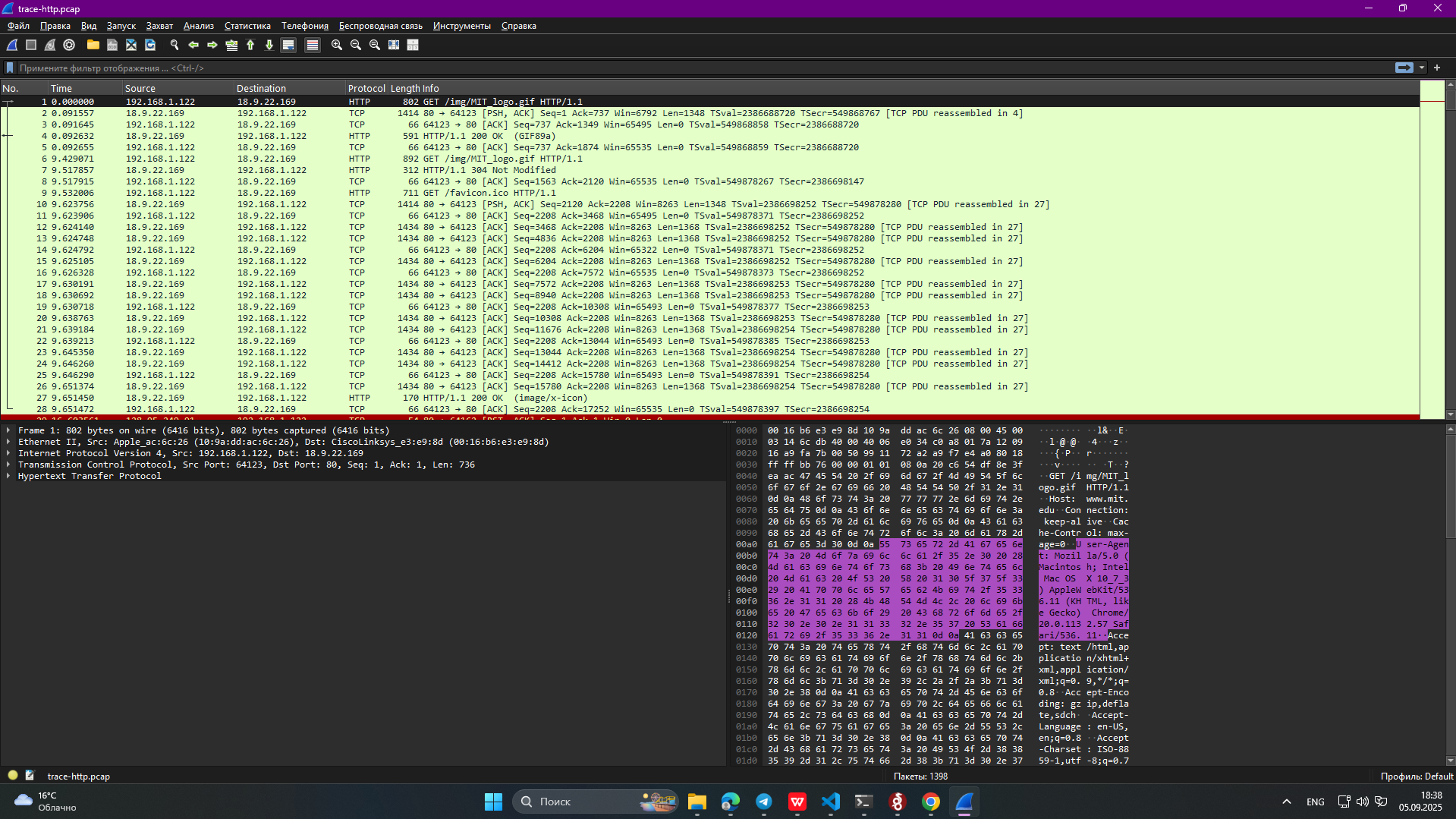


Рис. 1 открытие файла трассировки trace-http.pcap

• Количество пакетов, который были перехвачены Wireshark и присутствуют в этом файле? Ответ : 1398

Каждый пакет имеет временную метку и данные о кадре, кото рые говорят нам о времени создания пакета и о том, в каком именно кадре он передается. Другие данные включают адрес ис точника и назначения.

• Какой общий объем перехваченных пакетов (в байтах)? Ответ : 1223530

• Количество пакетов HTTP Request? Ответ : 98

• Количество пакетов HTTP Response? Ответ : 98

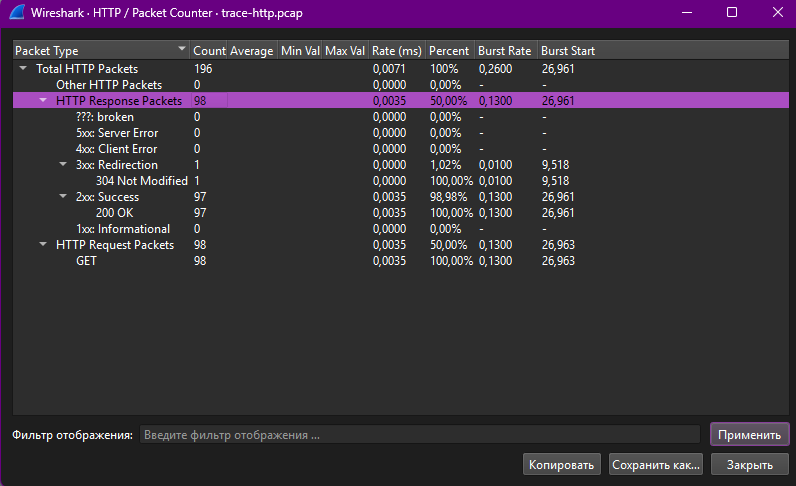


Рис. 2 Проверка статистики пакетов

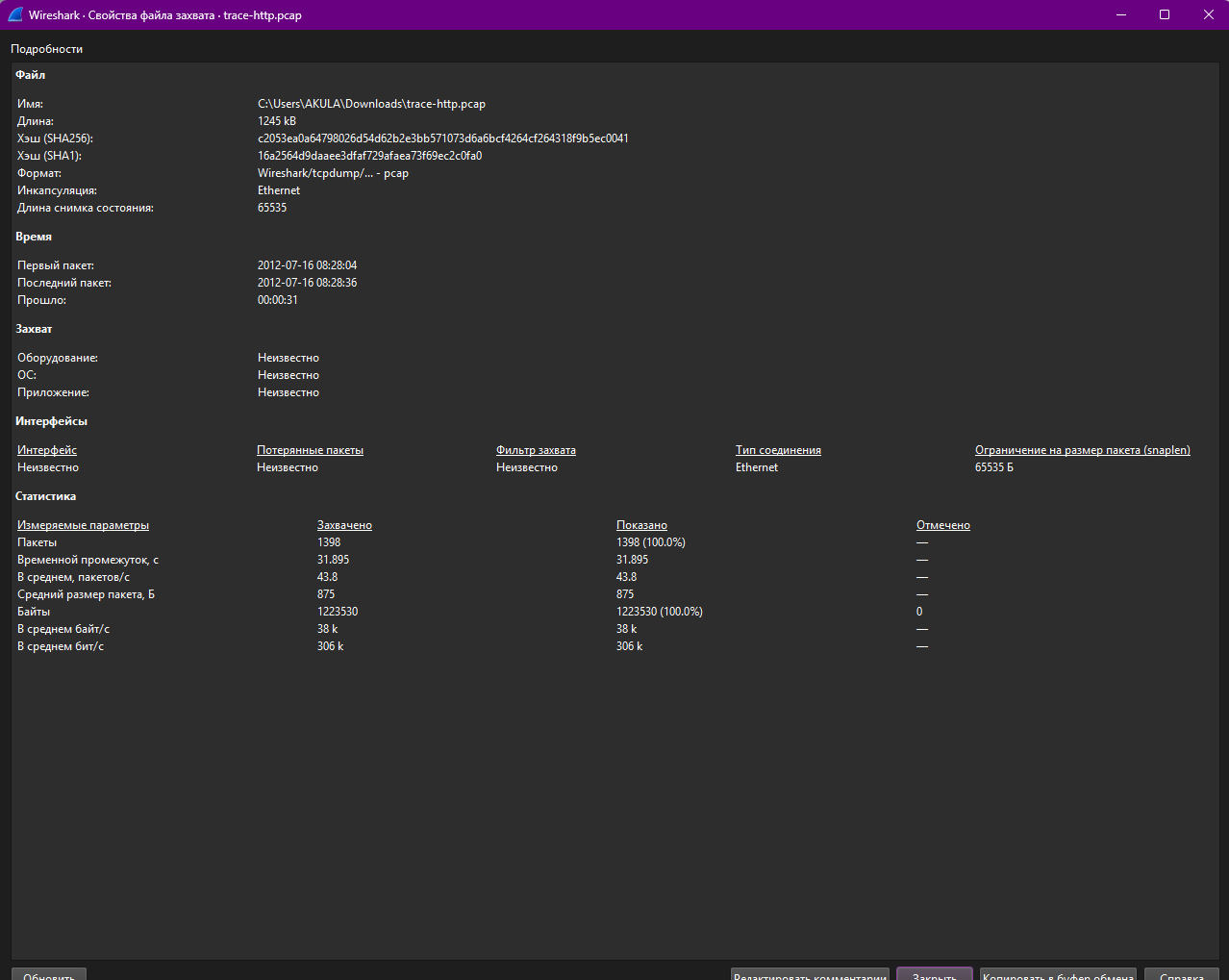
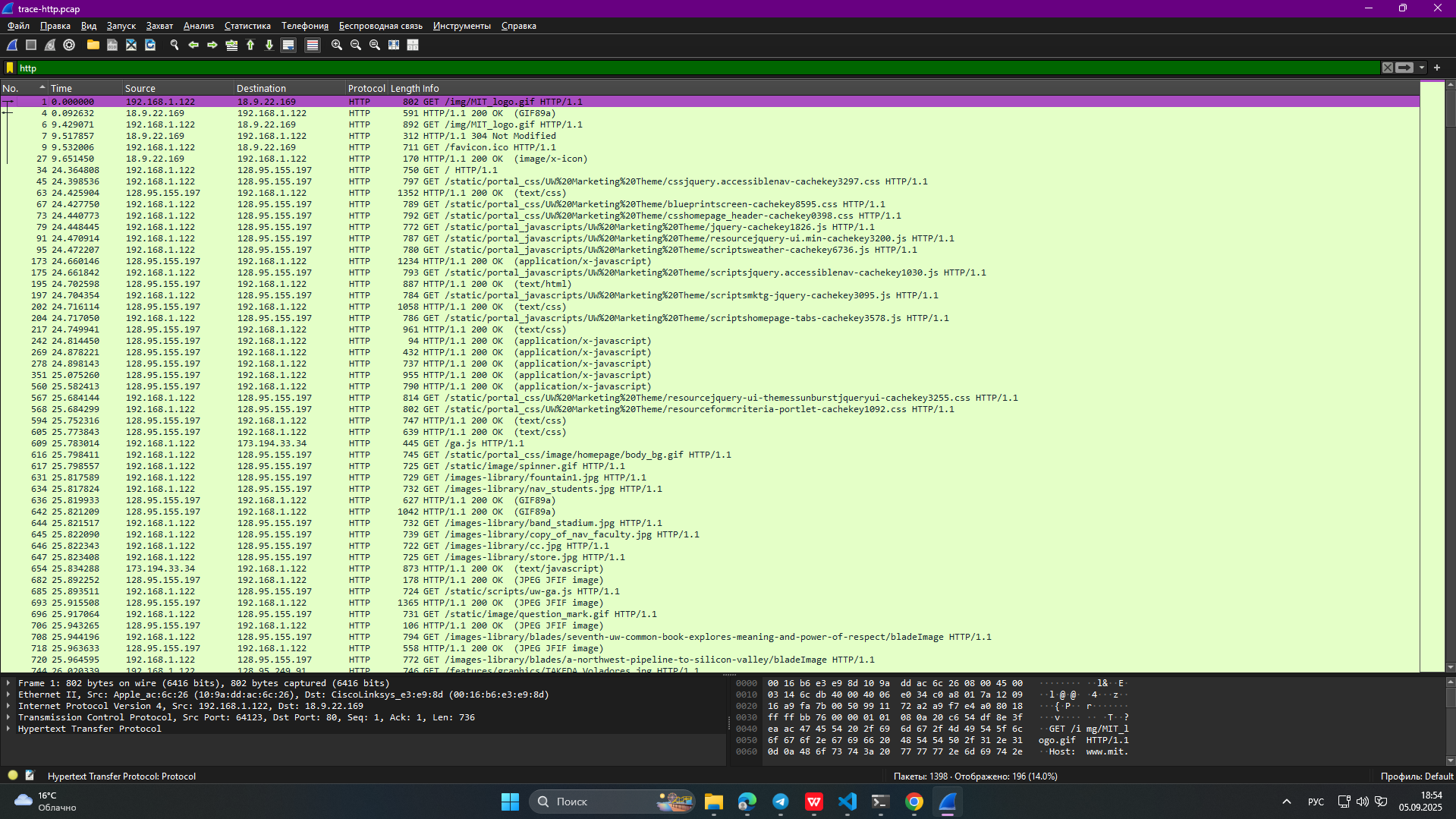
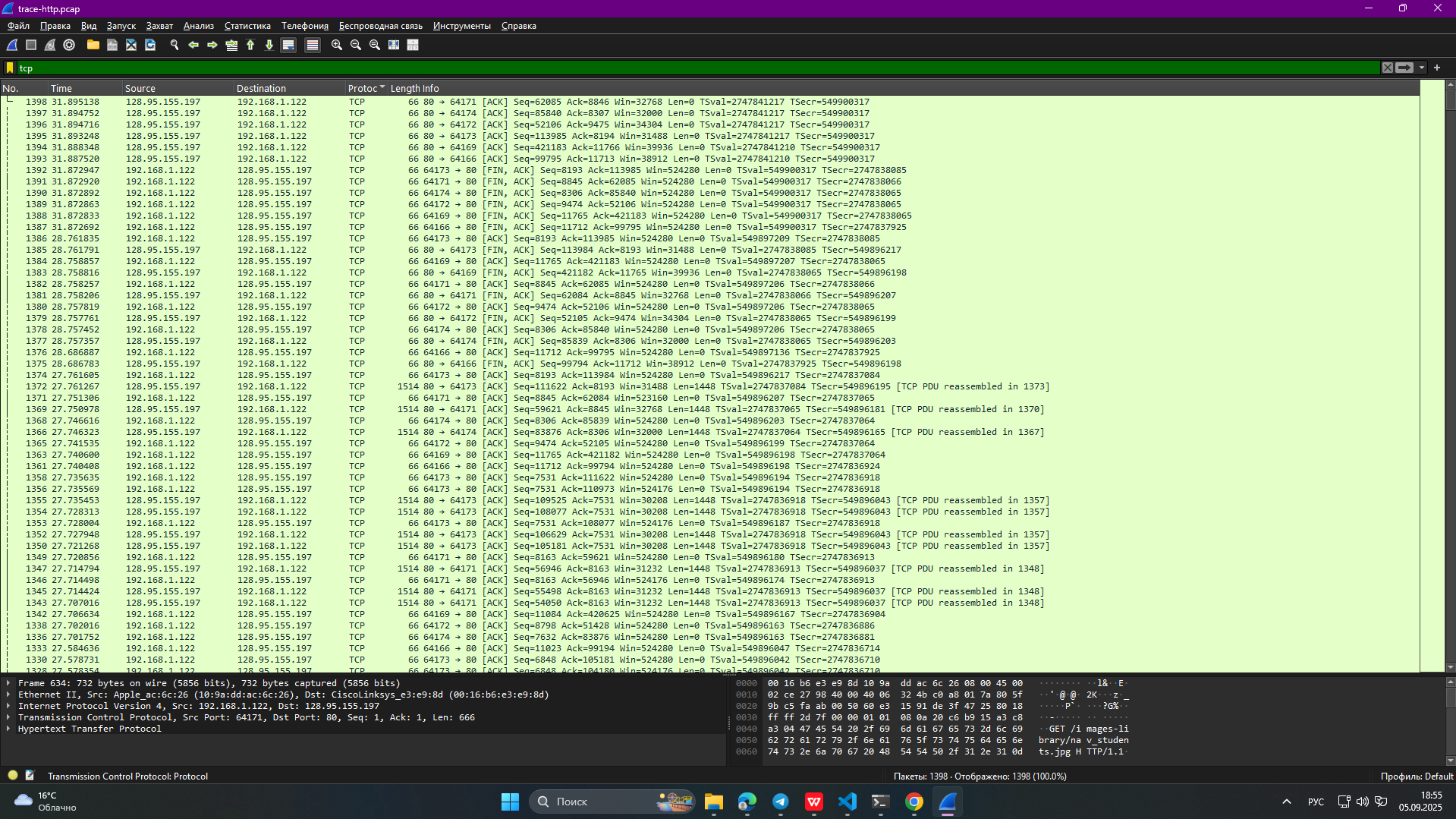


Рис. 3 Общая статистика сохранённого файла

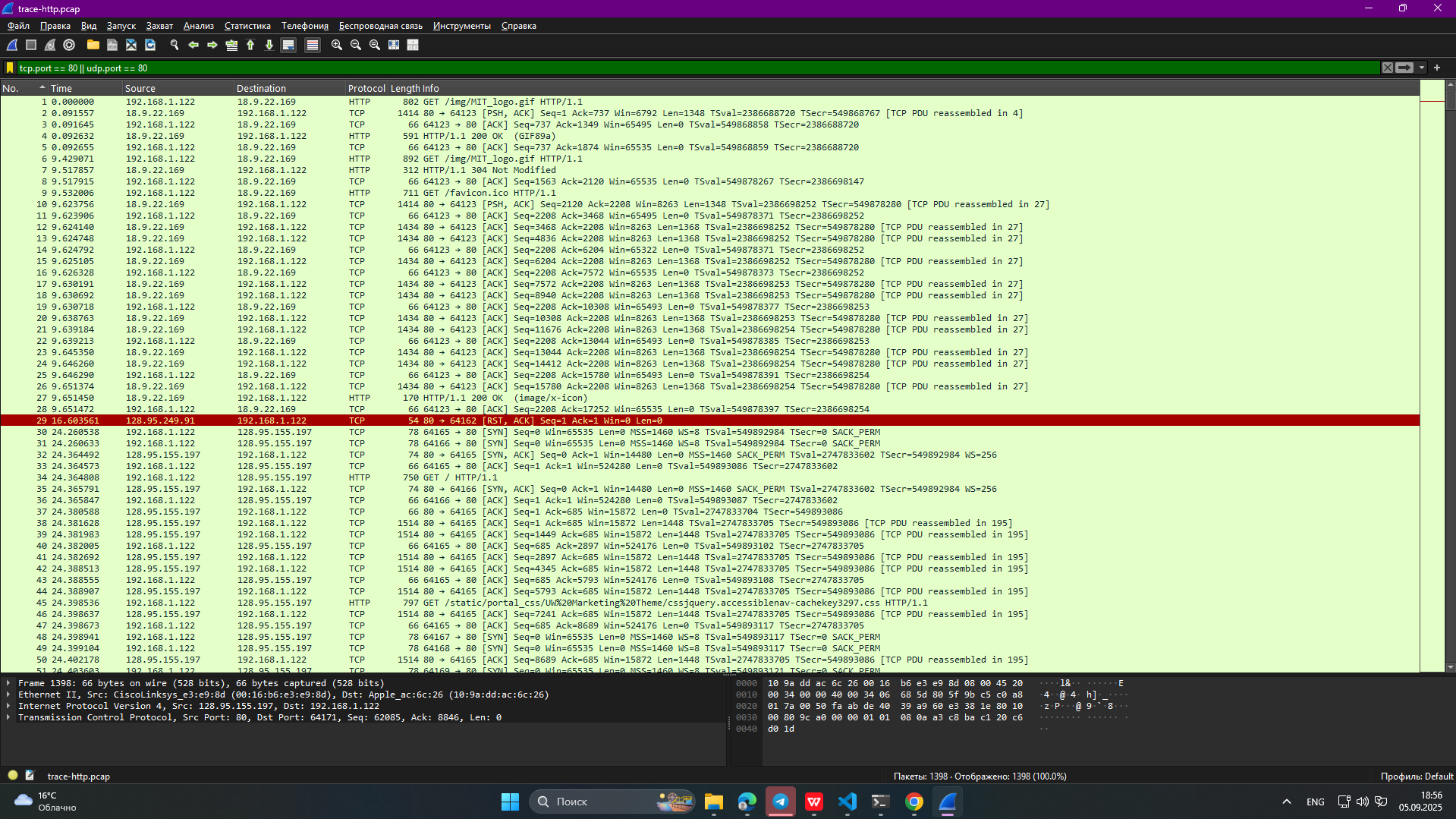
Задание 2. Изучение трассировки запроса и ответа



Применение фильтра для HTTP

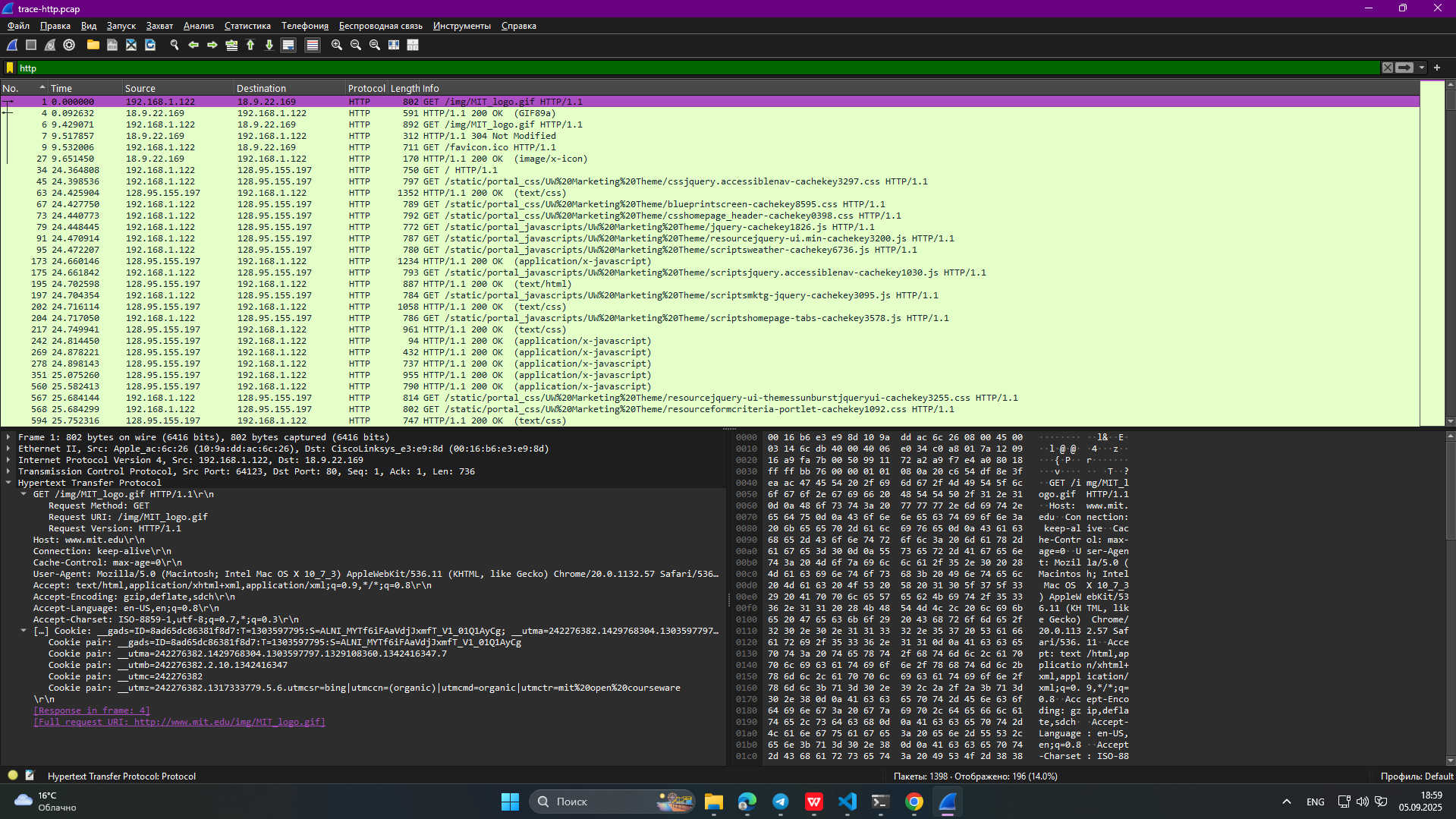


TCP



TCP или UDP с использованием порта 80

Просмотр GET запроса и изучение HTTP-заголовка



первый GET в трассировке и разверните его HTTP блок (это будет 802 GET /img/MIT\_logo.gif HTTP/1.1), как показано ниже).

Host: [www.mit.edu\r\n](http://www.mit.edu/r/n)

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_7\_3) AppleWebKit/536.11 (KHTML, like Gecko) Chrome/20.0.1132.57 Safari/536.11\r\n

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,\*/\*;q=0.8\r\n

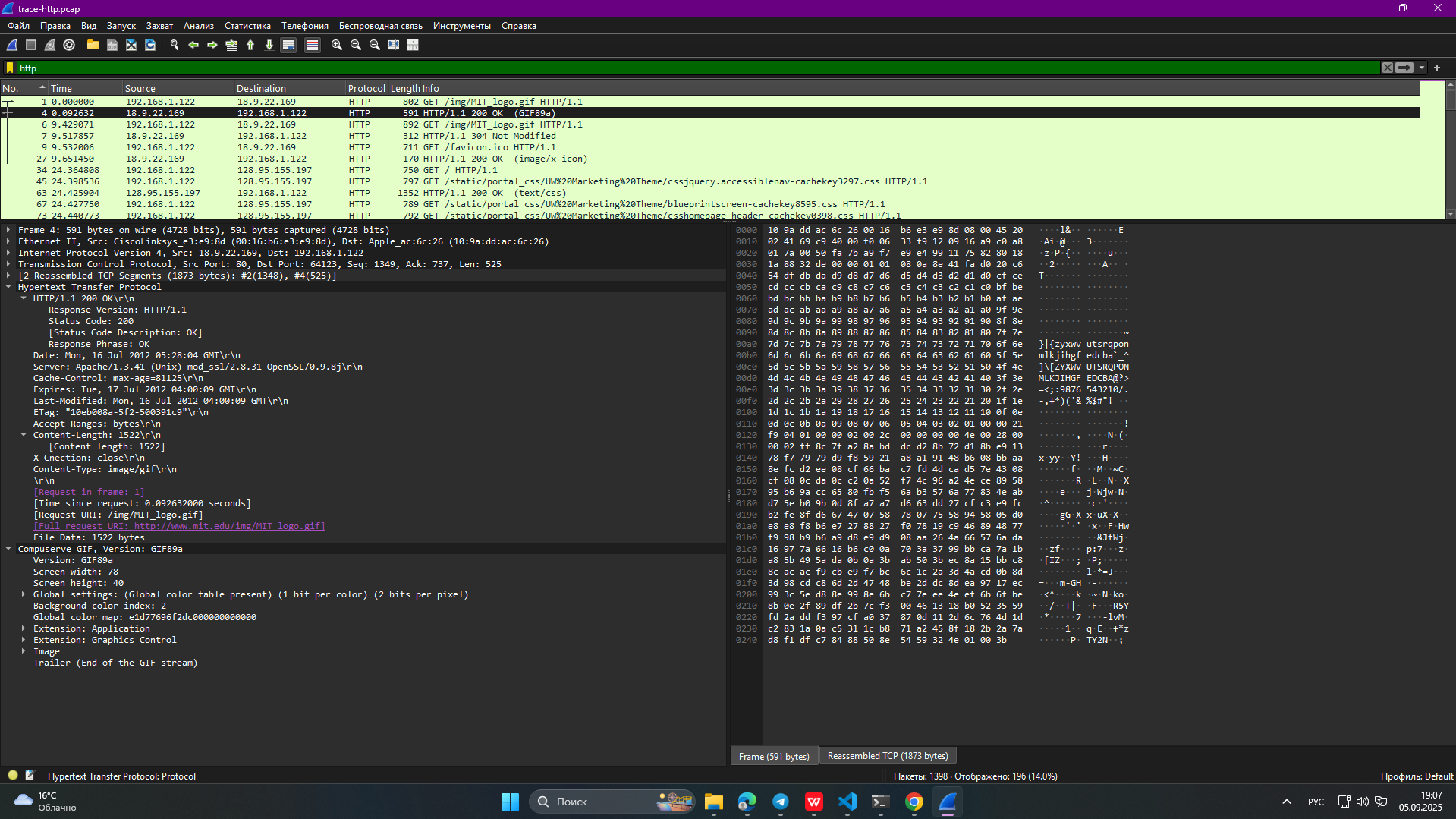
Accept-Encoding: gzip,deflate,sdch\r\n

Accept-Language: en-US,en;q=0.8\r\n

[…] Cookie: \_\_gads=ID=8ad65dc86381f8d7:T=1303597795:S=ALNI\_MYTf6iFAaVdjJxmfT\_V1\_01Q1AyCg; \_\_utma=242276382.1429768304.1303597797.1329108360.1342416347.7; \_\_utmb=242276382.2.10.1342416347; \_\_utmc=242276382; \_\_utmz=242276382.1317333779.5.6

Cache-Control: max-age=0\r\n

Заголовковки имеются



Изучение GET-запроса

Server: Apache/1.3.41 (Unix) mod\_ssl/2.8.31 OpenSSL/0.9.8j\r\n

Date: Mon, 16 Jul 2012 05:28:04 GMT\r\n

Last-Modified: Mon, 16 Jul 2012 04:00:09 GMT\r\n

Cache-Control: max-age=81125\r\n

Expires: Tue, 17 Jul 2012 04:00:09 GMT\r\n

ETag: "10eb008a-5f2-500391c9"\r\n

Вот подробный анализ HTTP-заголовков из вашего примера, а также объяснение, какую полезную информацию они содержат и что она означает.

1. Server

Пример: Server: Apache/1.3.41 (Unix) mod\_ssl/2.8.31 OpenSSL/0.9.8j

Описание: Этот заголовок указывает, какой веб-сервер обработал запрос. В данном случае:

Apache/1.3.41 — версия веб-сервера Apache.

Unix — операционная система, на которой работает сервер.

mod\_ssl/2.8.31 — модуль для поддержки SSL/TLS.

OpenSSL/0.9.8j — версия библиотеки OpenSSL, используемой для шифрования.

Полезная информация:

Позволяет определить, какое программное обеспечение используется на сервере.

Может быть полезно для диагностики совместимости или уязвимостей (например, если версия Apache или OpenSSL устарела).

2. Date

Пример: Date: Mon, 16 Jul 2012 05:28:04 GMT

Описание: Указывает дату и время, когда сервер сформировал и отправил ответ на запрос.

Полезная информация:

Помогает клиенту понять, насколько актуален ответ.

Используется для синхронизации времени между клиентом и сервером.

3. Last-Modified

Пример: Last-Modified: Mon, 16 Jul 2012 04:00:09 GMT

Описание: Указывает дату и время последнего изменения запрошенного ресурса (например, HTML-страницы, изображения).

Полезная информация:

Клиент может использовать этот заголовок для кэширования: если ресурс не изменился, сервер может вернуть ответ 304 Not Modified, экономя трафик.

Помогает определить, нужно ли загружать ресурс заново.

4. Cache-Control

Пример: Cache-Control: max-age=81125

Описание: Управляет кэшированием ответа. В данном случае:

max-age=81125 — ресурс можно кэшировать на 81125 секунд (примерно 22.5 часа).

Полезная информация:

Определяет, как долго клиент или промежуточные прокси могут хранить ответ в кэше.

Помогает снизить нагрузку на сервер и ускорить загрузку страниц для пользователей.

5. Expires

Пример: Expires: Tue, 17 Jul 2012 04:00:09 GMT

Описание: Указывает дату и время, после которого кэшированный ответ считается устаревшим.

Полезная информация:

Альтернативный способ управления кэшированием (часто используется вместе с Cache-Control).

Клиент не будет использовать кэшированный ответ после указанной даты.

6. ETag

Пример: ETag: "10eb008a-5f2-500391c9"

Описание: Уникальный идентификатор версии ресурса. Используется для проверки, изменился ли ресурс с момента последнего запроса.

Полезная информация:

Клиент может отправить этот идентификатор в следующем запросе (If-None-Match), чтобы сервер вернул ресурс только в случае его изменения.

Помогает избежать повторной передачи неизменённых данных.

Пример использования заголовков в GET-запросе

Представим, что клиент отправляет GET-запрос на страницу example.com/page.html.

Сервер возвращает ответ с указанными заголовками.

Клиент сохраняет ресурс в кэш и запоминает ETag и Last-Modified.

При следующем запросе клиент может отправить заголовки If-None-Match и If-Modified-Since.

Если ресурс не изменился, сервер вернёт 304 Not Modified, и клиент использует кэшированную версию, экономя трафик и время.