МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ПОВОЛЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра радиотехники и связи

**Отчет по практической работе №2**

«НАСТРОЙКА СРЕДЫ ВЕБ-РАЗРАБОТКИ»

Выполнил: ст. группы ИСТ-33

Лебедева А. О.

Проверил: доцент, к.т.н

Чесноков С. Е.

Дата защиты \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Йошкар-Ола

2025

Оглавление

[Введение: 3](#_Toc211814485)

[Задание 1. Загрузка файла трассировки http 4](#_Toc211814486)

[Задание 2. Изучение трассировки запроса и ответа 7](#_Toc211814487)

[Задание 3. Кэширование контента. 12](#_Toc211814488)

[Задание 4. Сложные страницы. 14](#_Toc211814489)

[Контрольные вопросы: 18](#_Toc211814490)

# ****Введение:****

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 идентифицировать 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

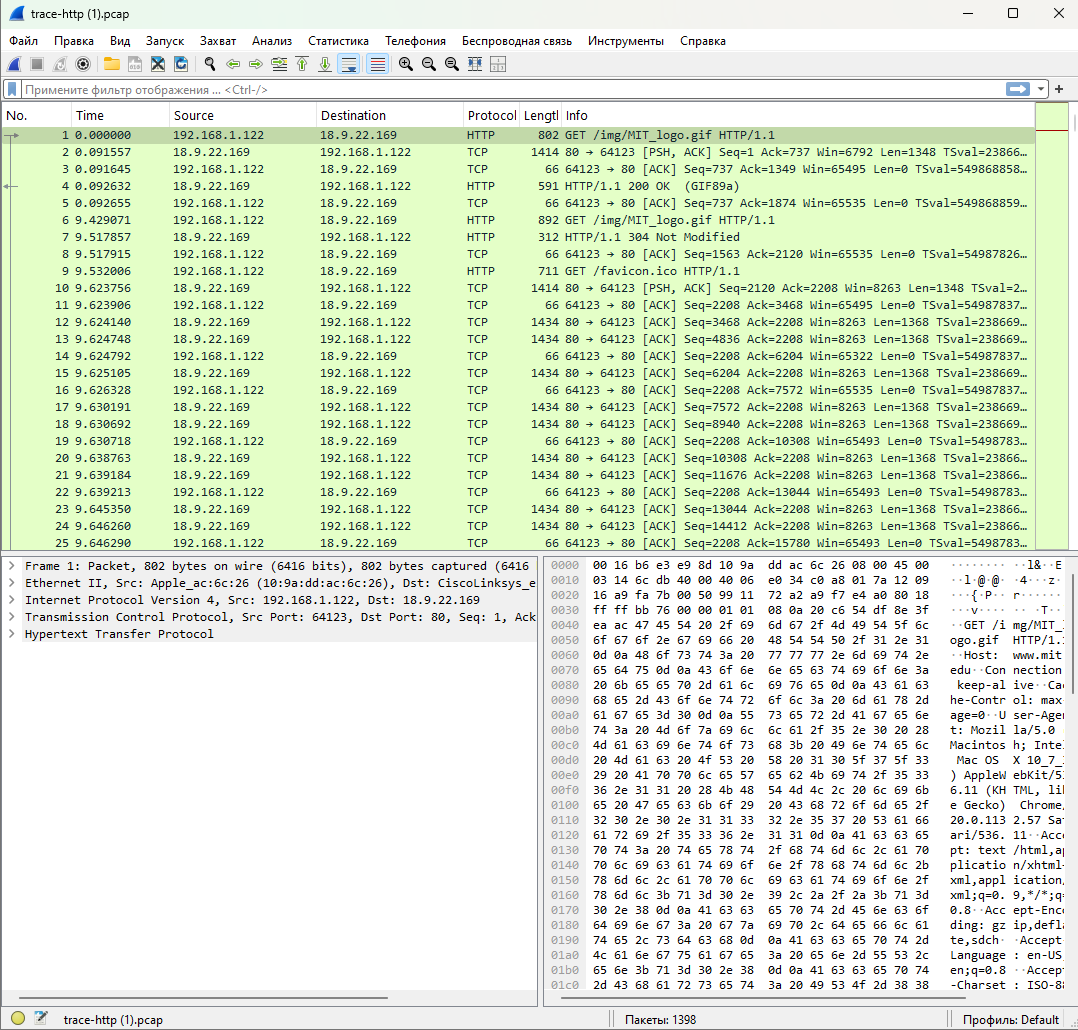


Рисунок 1: Сохраненный файл трассировки

**Ответы на вопросы:**

1. Количество пакетов, который были перехвачены Wireshark и

присутствуют в этом файле?

**1398**

1. Какой общий объем перехваченных пакетов (в байтах)?

**1223530**

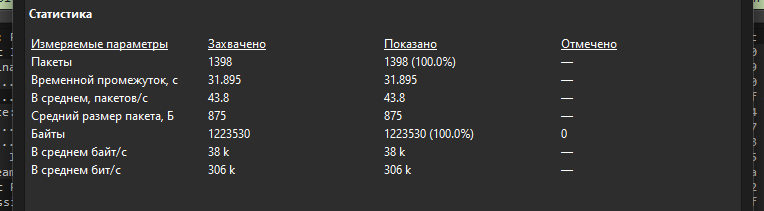
****

Рисунок 2: количество пакетов и общий объем перехваченных пакетов

1. Количество пакетов HTTP Request?  
   Применяем фильтр для HTTP-запросов, смотрим статистику.

**98**

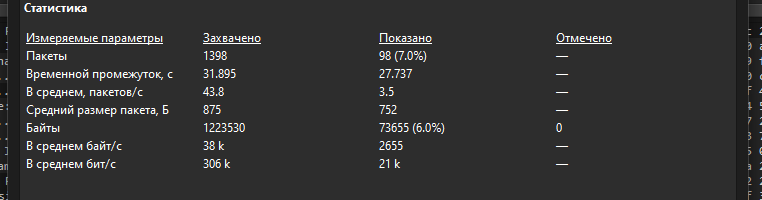


Рисунок 3: Количество пакетов HTTP Request

1. Количество пакетов HTTP Response?

**98**Применяем фильтр, смотрим количество пакетов в статистике или в правом нижнем углу.

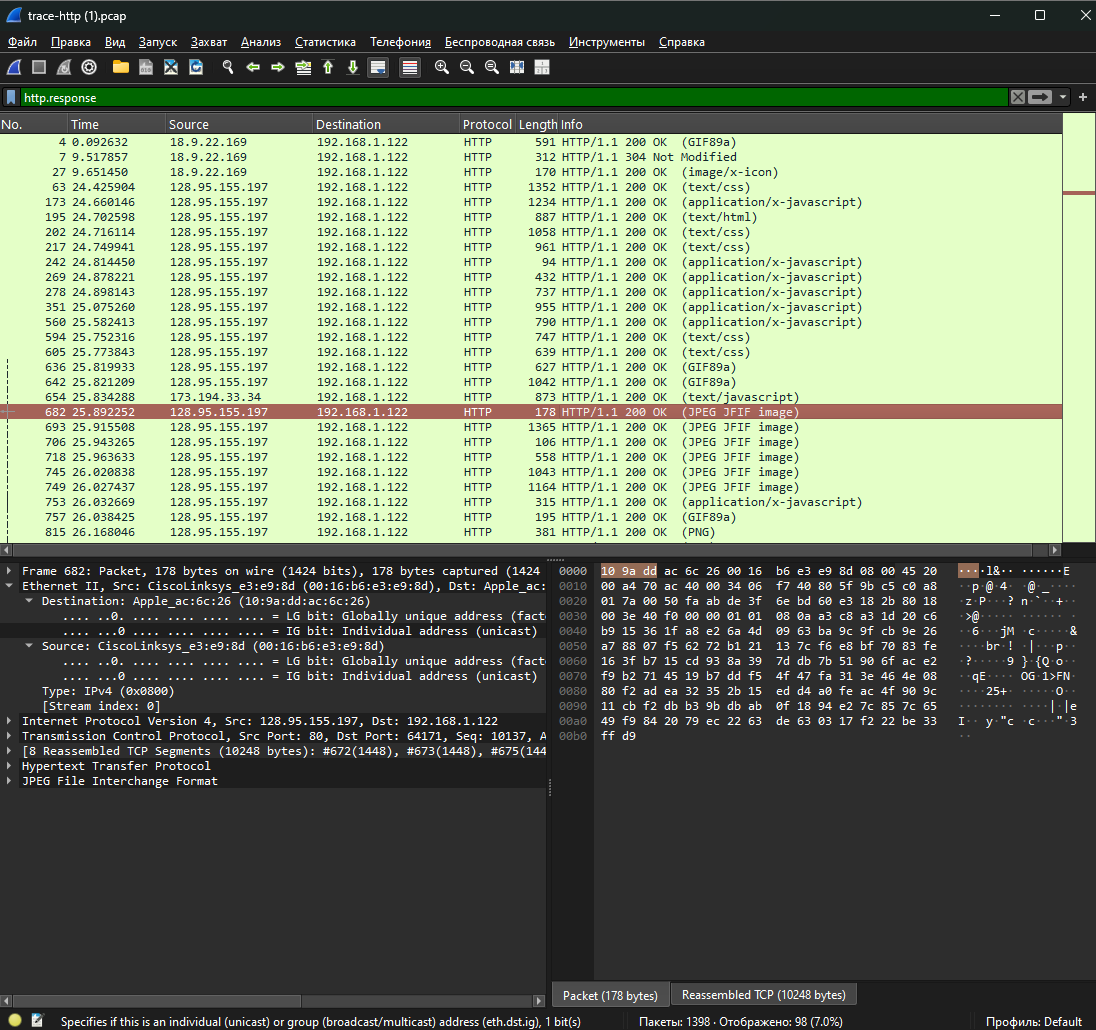


Рисунок 4**:** Количество пакетов HTTP Response

# Задание 2. Изучение трассировки запроса и ответа

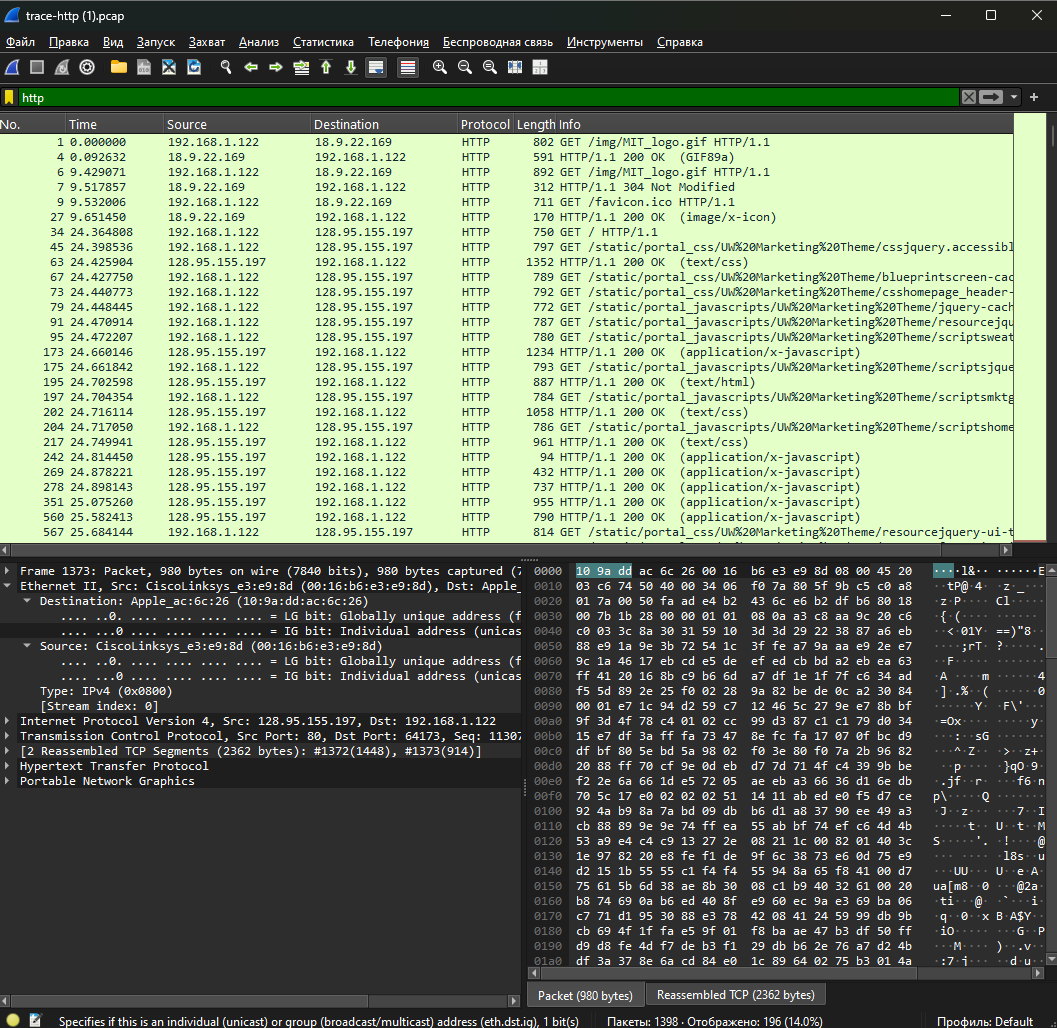


Рисунок 5: Применение фильтра «http»

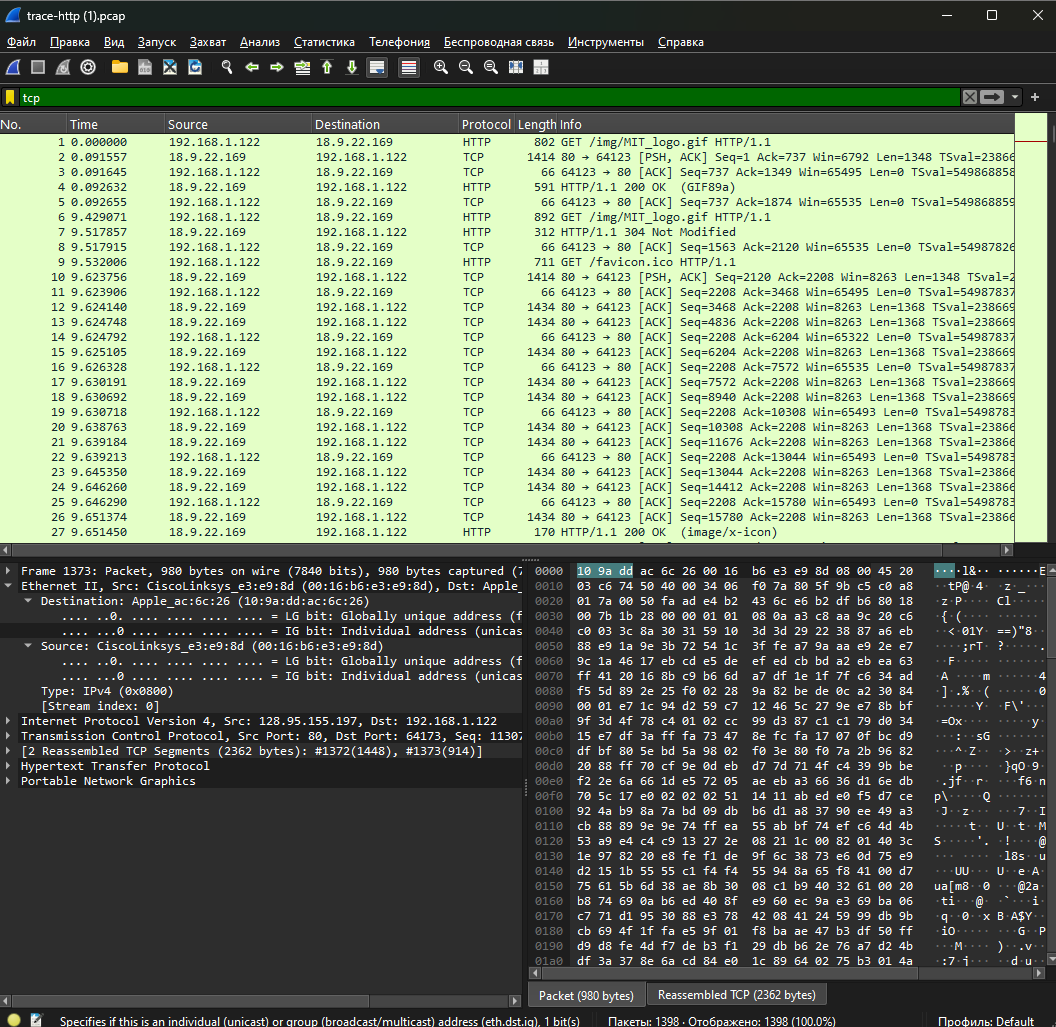
****

Рисунок 6: Применение фильтра «tcp»

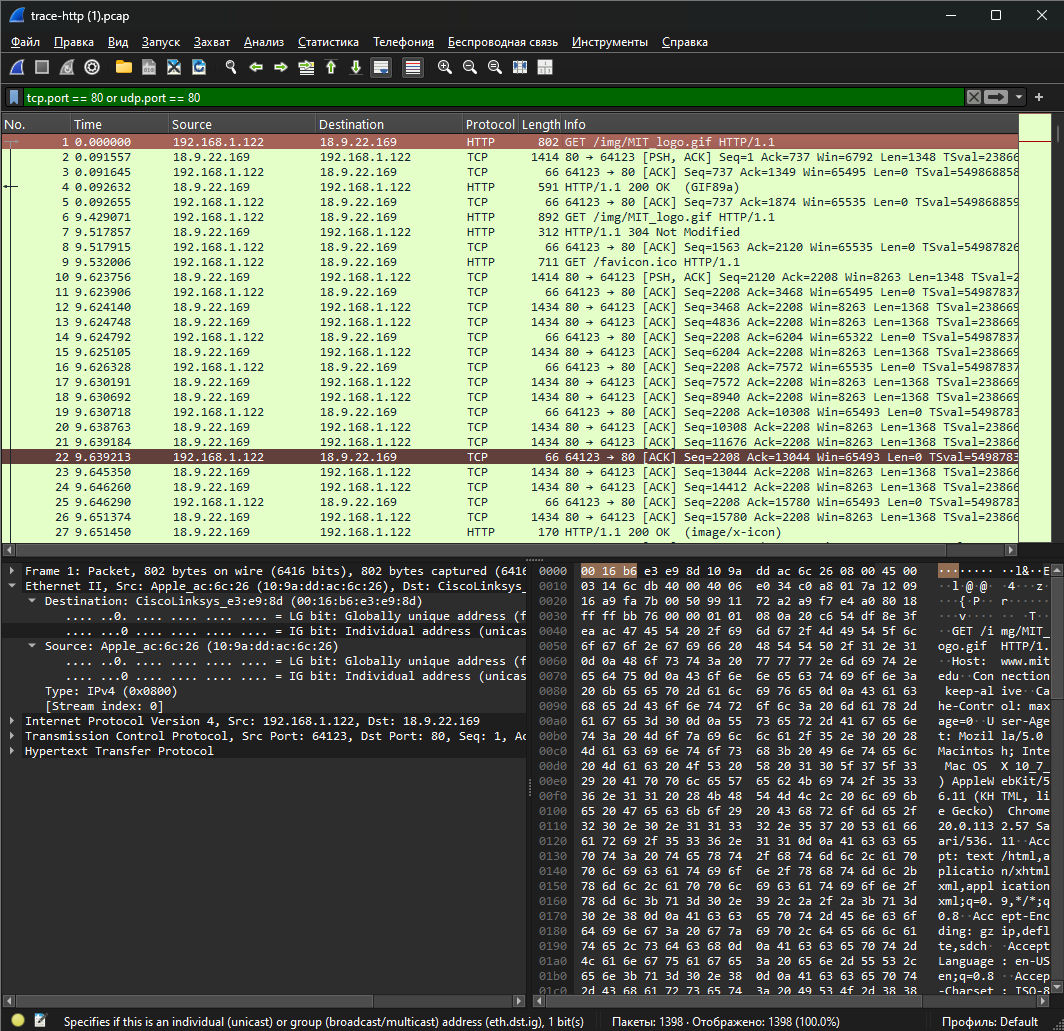
****

Рисунок 7: Применение фильтра TCP или UDP с использованием порта 80

**Найденные заголовки:**

* **Request Method:** метод запроса (клиент запрашивает ресурс).
* **Request URI:** путь к файлу, который запрашивается.
* **Request Version:** версия протокола HTTP.
* **Host:** указывает, к какому хосту отправлен запрос (нужно при виртуальном хостинге).
* **Connection:** просит сервер не закрывать соединение после ответа, чтобы использовать его повторно.
* **Cache-Control:** инструкция, как кэшировать контент (здесь — не использовать кэш).
* **User-Agent:** идентифицирует браузер и операционную систему клиента.
* **Accept:** какие типы данных клиент готов принять в ответе.
* **Accept-Encoding:** какие алгоритмы сжатия клиент поддерживает.
* **Accept-Language:** предпочитаемые языки клиента.
* **Accept-Charset:** какие кодировки поддерживает клиент.
* **Cookie:** данные сессии, передаваемые браузером серверу.

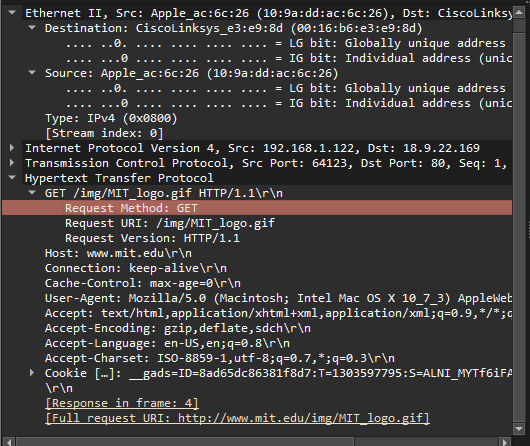
****

Рисунок 8: Найденные заголовки

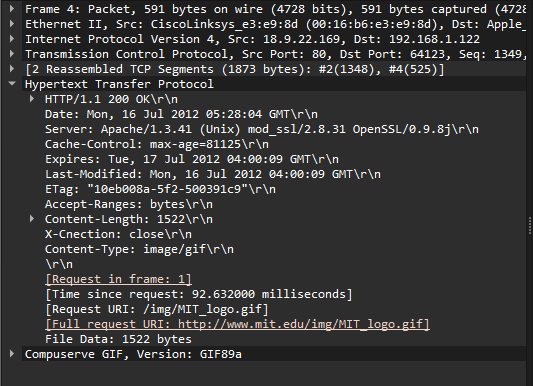
****

Рисунок 9: Просмотр GET ответа

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

Показывает программное обеспечение, на котором работает сервер:

Apache/1.3.41 — тип и версия веб-сервера.

mod\_ssl / OpenSSL — говорит, что сервер поддерживает защищённые соединения (SSL/TLS).

Помогает определить ПО и возможности сервера. Часто скрывается по соображениям безопасности.

1. Date: Mon, 16 Jul 2012 05:28:04 GMT

Указывает время, когда сервер сформировал ответ.

Можно понять момент обработки запроса сервером и проверить задержку между запросом и ответом.

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

Дата и время последнего изменения ресурса на сервере.

Браузер может использовать её, чтобы не скачивать файл, если он не изменился.

1. Cache-Control: max-age=81125

Указывает, как долго браузер может хранить этот ресурс в кэше.

max-age=81125 означает, что ответ можно хранить 81 125 секунд (примерно 22,5 часа).

Уменьшает количество повторных загрузок и экономит трафик.

1. Expires: Tue, 17 Jul 2012 04:00:09 GMT

Дата, когда ресурс считается "устаревшим".

До этого времени браузер может использовать кэшированную копию без повторного запроса к серверу.

Помогает оптимизировать кэширование и ускорить загрузку сайта.

1. ETag: "10e0b808a-5f2e-500391c9"

Уникальный идентификатор версии ресурса.

Если содержимое файла изменится, сервер сгенерирует новый ETag.

Позволяет клиенту запрашивать только обновлённые файлы, экономит трафик и ускоряет загрузку.

# Задание 3. Кэширование контента.



Рисунок 10: Скрин загруженного Gif изображения, по которому обрабатывался запрос

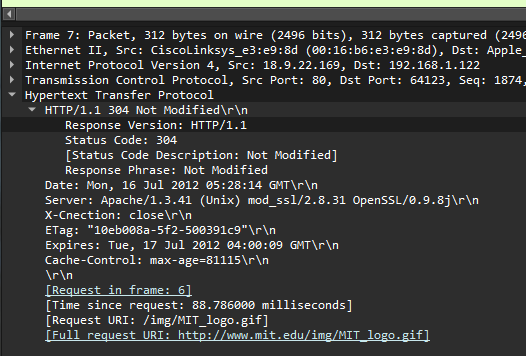


Рисунок 11: Ответ на повторную выборку

На данном скриншоте представлен ответ сервера с кодом состояния **HTTP/1.1 304 Not Modified**. Этот код означает, что запрашиваемый браузером файл не изменился с момента последней загрузки, и сервер сообщает, что можно использовать кэшированную копию, не загружая контент повторно.

1. **Date** указывает время формирования ответа сервером — Mon, 16 Jul 2012 05:28:14 GMT.
2. **Server** содержит информацию о типе и версии веб-сервера: Apache/1.3.41 (Unix) mod\_ssl/2.8.31 OpenSSL/0.9.8j, что говорит о том, что сервер работает под управлением Apache с поддержкой SSL-шифрования.
3. **ETag** ("10eb808a-5f2-500391c9") представляет собой уникальный идентификатор версии файла. Он используется сервером, чтобы определить, изменилась ли версия ресурса.
4. **Expires** и **Cache-Control** (max-age=81125) сообщают браузеру, как долго можно хранить кэшированную копию файла без повторной проверки на сервере. Эти параметры управляют временем жизни кэша.
5. **X-Cnection: close** указывает, что после отправки данного ответа соединение с клиентом будет закрыто.

# Задание 4. Сложные страницы.

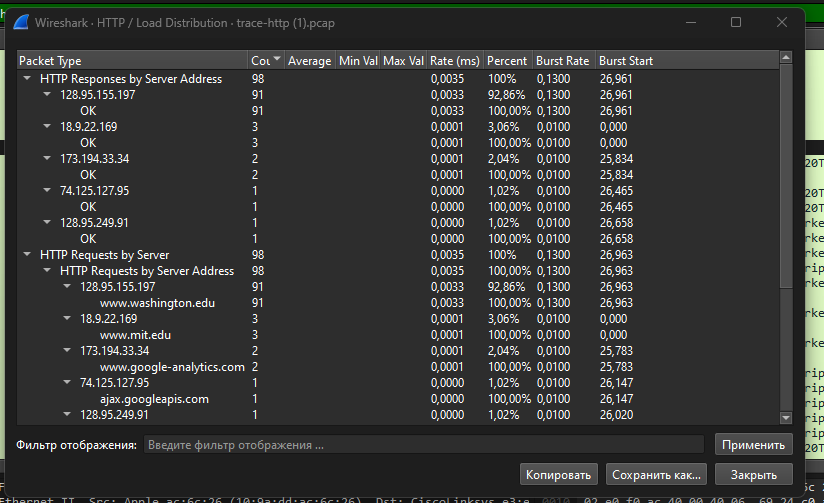


Рисунок 12: Отображение статистики по HTTP

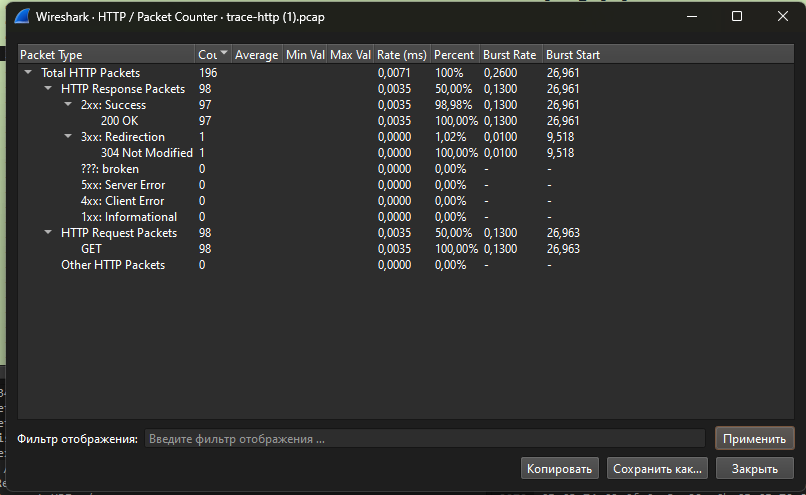


Рисунок 13: Отображение подсчета пакетов

Результаты тестирования:

Мобильные устройства

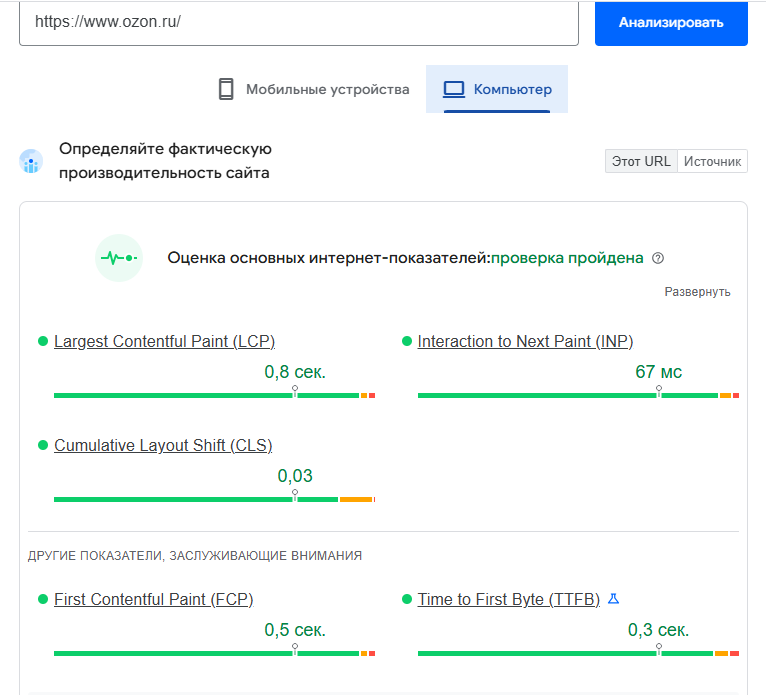


Рисунок 14: Тестирование 1

|  |  |  |
| --- | --- | --- |
| **Показатель** | **Значение** | **Интерпретация** |
| **Largest Contentful Paint (LCP)** | 1,2 сек. | Отлично (до 2,5 сек. — норма) |
| **Interaction to Next Paint (INP)** | 169 мс | Отлично (до 200 мс — норма) |
| **Cumulative Layout Shift (CLS)** | 0,33 | Плохой показатель (норма < 0,1) |
| **First Contentful Paint (FCP)** | 0,9 сек. | Отлично |
| **Time to First Byte (TTFB)** | 0,4 сек. | Отлично |
| **Итоговая оценка** | Проверка не пройдена | Есть проблемы с визуальной стабильностью (CLS) |

Компьютерная версия:

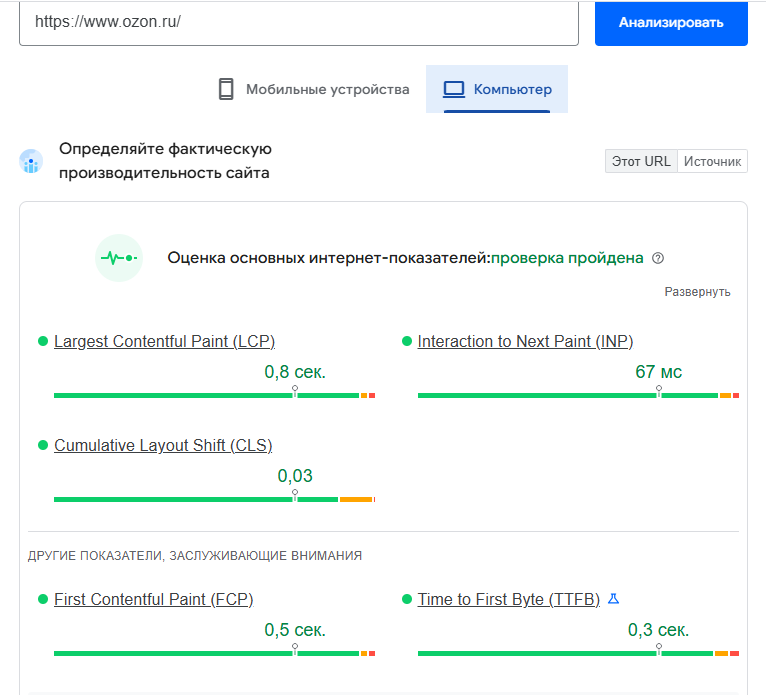


Рисунок 15: Тестирование 2

|  |  |  |
| --- | --- | --- |
| **Показатель** | **Значение** | **Интерпретация** |
| **Largest Contentful Paint (LCP)** | 0,8 сек. | Отлично |
| **Interaction to Next Paint (INP)** | 67 мс | Отлично |
| **Cumulative Layout Shift (CLS)** | 0,03 | Отлично |
| **First Contentful Paint (FCP)** | 0,5 сек. | Отлично |
| **Time to First Byte (TTFB)** | 0,3 сек. | Отлично |
| **Итоговая оценка** | Проверка пройдена | Высокая производительность |

Вывод:

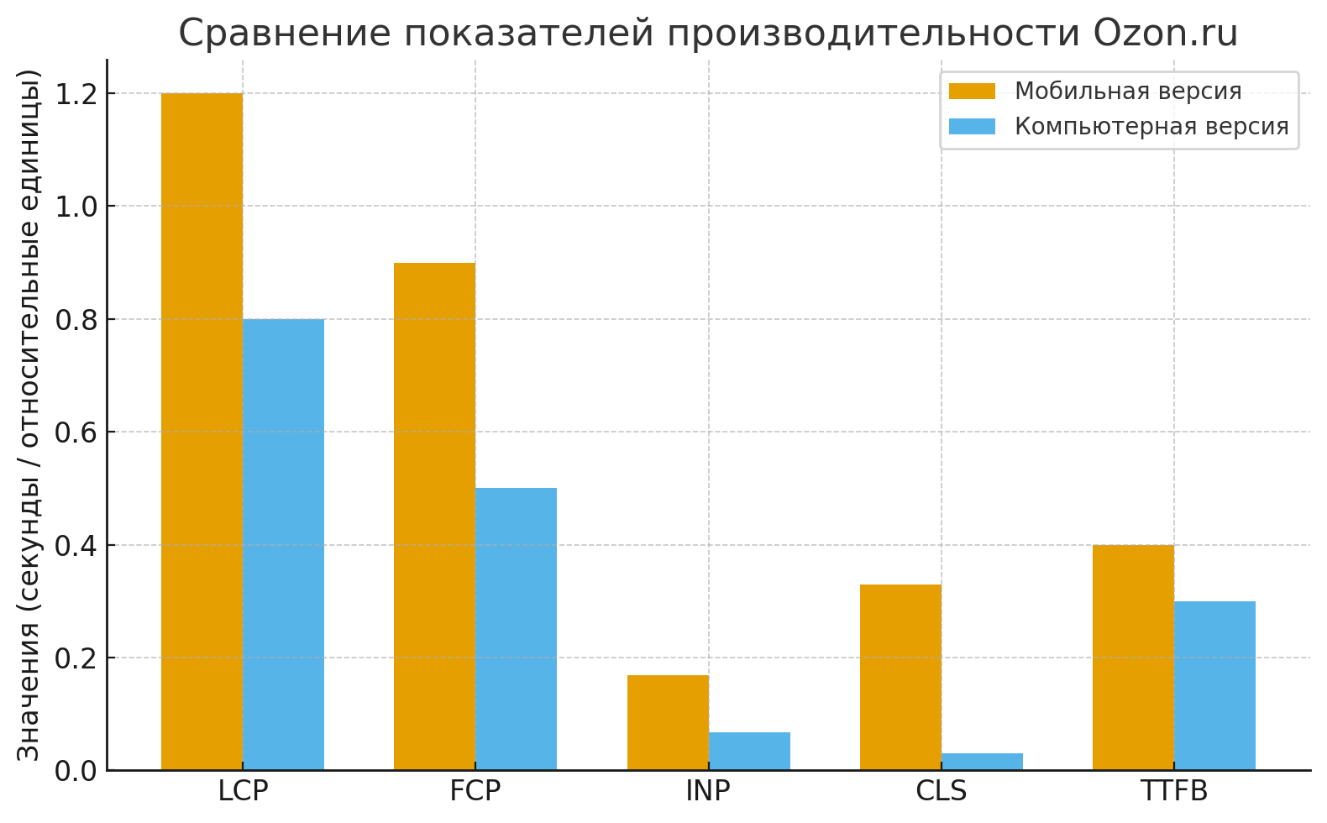


Рисунок 16: Гистограмма

**Мобильная версия** сайта показывает высокую скорость отклика и загрузки, но страдает от визуальной нестабильности (высокий CLS = 0,33). Это может быть связано с подгрузкой рекламы или динамических баннеров.

**Компьютерная версия** демонстрирует отличные результаты по всем метрикам: быстрая загрузка, мгновенный отклик и стабильная структура страницы.

# Контрольные вопросы:

1. Что такое протокол HTTP и для чего он используется?

Протокол HTTP (HyperText Transfer Protocol) — это протокол прикладного уровня, предназначенный для передачи данных в распределённых информационных системах, таких как Всемирная паутина. Он работает по схеме «клиент-сервер» и используется для обмена запросами и ответами между веб-браузером (клиентом) и веб-сервером.

Основные функции HTTP:

* Передача веб-ресурсов: HTML-страниц, изображений, CSS, JavaScript и других файлов.
* Текстовая природа: Заголовки запросов и ответов представлены в текстовом формате, что упрощает анализ и отладку.
* Работа поверх TCP/IP: По умолчанию использует порт 80.
* Поддержка HTTPS: Зашифрованная версия протокола, обеспечивающая конфиденциальность данных.

1. Каковы основные методы HTTP и какова их функция?

Основные методы HTTP — это команды, которые определяют действие, которое клиент хочет выполнить с указанным ресурсом на сервере. Согласно документу, в HTTP/1.1 поддерживаются следующие методы:

Основные методы HTTP и их функции:

* GET - используется для получения информации с данного сервера с использованием данного URI. Запросы, использующие GET, должны извлекать только данные и не должны оказывать никакого другого воздействия на данные.
* HEAD - то же, что и GET, но он передает только строку состояния и раздел заголовка.
* POST - используется для отправки данных на сервер, например, информации о клиенте, загрузки файла и т.д. С использованием HTML-форм.
* PUT - заменяет все текущие представления целевого ресурса загруженным контентом.
* DELETE - удаляет все текущие представления целевого ресурса, заданные URI.
* CONNECT - устанавливает туннель к серверу, идентифицированному заданным URI.
* OPTIONS - описание параметров связи для целевого ресурса.
* TRACE - выполняет проверку обратного цикла сообщения вместе с указанием пути к целевому ресурсу

Важно: Все методы HTTP чувствительны к регистру и должны указываться в верхнем регистре.

1. Объясните разницу между методами GET и POST.

Основное различие между методами GET и POST заключается в их назначении:

* GET предназначен для получения данных с сервера. Он используется, когда нужно просто запросить информацию, например, загрузить веб-страницу, изображение или любой другой ресурс. Согласно документу, "запросы, использующие GET, должны извлекать только данные и не должны оказывать никакого другого воздействия на данные". Это делает его безопасным и повторяемым (идемпотентным) методом.
* POST предназначен для отправки данных на сервер для обработки. Он используется, когда нужно что-то изменить на сервере: отправить комментарий, добавить товар в корзину, загрузить файл или передать данные формы. Как указано в документе, он используется для "отправки данных на сервер, например, информации о клиенте, загрузки файла и т.д.".

1. Что такое статус-коды HTTP и какова их роль в протоколе?

Статус-коды HTTP (HTTP Status Codes) — это трехзначные числовые коды, которые сервер отправляет в ответ на HTTP-запрос клиента. Они являются неотъемлемой частью строки состояния (Status-Line) HTTP-ответа.

Их роль в протоколе заключается в том, чтобы быстро и стандартизированно проинформировать клиента о результате обработки его запроса.

1. Перечислите основные статус-коды HTTP и объясните их значение.

Статус-коды сгруппированы по классам, где первая цифра определяет категорию ответа. Вот ключевые коды из каждого класса:

Класс 2xx: Успех (Success)

* 200 OK: Запрос успешно обработан. Это стандартный ответ для успешных операций GET, HEAD или POST.
* 201 Created: Запрос успешно выполнен и в результате был создан новый ресурс (часто после PUT или POST).
* 204 No Content: Сервер успешно обработал запрос, но не возвращает никакого содержимого в теле ответа.

Класс 3xx: Перенаправление (Redirection)

* 301 Moved Permanently: Запрошенный ресурс был окончательно перемещен на новый URL, указанный в заголовке Location. Браузеры и поисковые системы обновляют свои ссылки.
* 302 Found: Ресурс временно доступен по другому URL. Браузеры перенаправляются, но продолжают использовать оригинальный URL для будущих запросов.
* 304 Not Modified: Используется для кэширования. Сервер сообщает клиенту, что ресурс не изменился с момента последнего запроса, и клиент может использовать свою кэшированную копию. Тело ответа не передается.

Класс 4xx: Ошибка клиента (Client Error)

* 400 Bad Request: Сервер не может обработать запрос из-за синтаксической ошибки клиента (например, неверный формат запроса).
* 401 Unauthorized: Для доступа к ресурсу требуется аутентификация. Клиент должен представиться с правильными учетными данными.
* 403 Forbidden: Сервер понял запрос, но отказывается его авторизовать. Доступ запрещен, даже при аутентификации.
* 404 Not Found: Самая известная ошибка. Сервер не может найти запрошенный ресурс.
* 405 Method Not Allowed: Метод (GET, POST и т.д.), указанный в запросе, не поддерживается сервером для данного ресурса.

Класс 5xx: Ошибка сервера (Server Error)

* 500 Internal Server Error: Общий статус для непредвиденных сбоев на стороне сервера, когда он не может более точно указать причину ошибки.
* 502 Bad Gateway: Сервер, выступая в роли шлюза или прокси, получил недействительный ответ от вышестоящего сервера.
* 503 Service Unavailable: Сервер временно не может обработать запрос из-за перегрузки или технического обслуживания.
* 504 Gateway Timeout: Сервер, acting as a gateway or proxy, did not receive a timely response from an upstream server.

1. Что такое заголовки HTTP и какую информацию они содержат?

Заголовки HTTP (HTTP Headers) — это пары "имя-значение", которые передаются в HTTP-запросе и ответе. Они не являются частью тела сообщения (например, HTML-кода страницы или данных формы), а содержат метаинформацию о запросе или ответе.

Их роль — управлять передачей данных, определять параметры обработки и предоставлять контекст для клиента и сервера.

Заголовки можно разделить на несколько категорий в зависимости от той информации, которую они несут:

1. Общие заголовки (General Headers)

Содержат информацию, относящуюся как к запросу, так и к ответу:

* Cache-Control: Директивы для кэширования (как долго можно хранить ресурс).
* Connection: Управление сетевым соединением (например, keep-alive).
* Date: Дата и время создания сообщения.

2. Заголовки запроса (Request Headers)

Содержат информацию о клиенте и о том, какой ответ он предпочитает.

* Host: Обязательный заголовок, указывает имя сервера и порт.
* User-Agent: Описывает тип и версию браузера/клиента.
* Accept, Accept-Language, Accept-Encoding: Сообщают серверу, какие типы контента, языки и кодировки понимает клиент.
* Cookie: Передает серверу данные cookies, хранящиеся на стороне клиента.
* Authorization: Содержит учетные данные для аутентификации на сервере.

3. Заголовки ответа (Response Headers)

Содержат дополнительную информацию об ответе сервера, которую нельзя поместить в строку состояния.

* Server: Информация о типе и версии сервера.
* Set-Cookie: Команда клиенту установить cookie.
* Location: Используется для перенаправления (редиректа), указывает новый адрес ресурса.
* Content-Type: Указывает MIME-тип данных в теле ответа (например, text/html, image/png).
* Content-Length: Сообщает размер тела ответа в байтах.

4. Заголовки сущности (Entity Headers)

Описывают содержимое (сущность) тела сообщения.

* Content-Type, Content-Length, Content-Encoding (например, gzip).

Таким образом, заголовки HTTP действуют как система управления и контекста для всего обмена данными по протоколу HTTP, определяя, как обрабатывать передаваемую информацию.

1. Какова роль заголовка User-Agent в HTTP-запросах?

Роль заголовка User-Agent - сообщить серверу информацию о клиентском приложении, которое выполняет запрос. Обычно это веб-браузер, но может быть и поисковый робот, или другое приложение.

1. Что такое HTTPS и как он отличается от HTTP?

HTTPS (HyperText Transfer Protocol Secure) — это безопасная, зашифрованная версия протокола HTTP. HTTPS обеспечивает конфиденциальность, целостность и аутентификацию передаваемых данных. Протокол HTTPS на самом деле представляет собой два протокола, работающих друг над другом. Первый протокол - это протокол безопасности, такой как SSL или TLS. Второй протокол, работающий поверх этого протокола безопасности, — это HTTP.

1. Как работает механизм кэширования в HTTP?

Как это работает:

**Первый запрос**: когда браузер впервые запрашивает ресурс (например, GET /img/logo.gif), сервер возвращает его со статусом 200 OK и включает заголовки для кэширования, например:

* Last-Modified (дата последнего изменения файла на сервере).
* ETag (уникальный идентификатор версии файла).

Браузер сохраняет ресурс в своем кэше вместе с этими метаданными.

**Повторный запрос**: при следующем посещении страницы, браузер хочет снова загрузить тот же ресурс. Вместо простого GET, он отправляет условный запрос, включая заголовки:

* If-Modified-Since: [дата из Last-Modified] или If-None-Match: [значение из ETag]

Это означает: "Пришли ресурс, только если он изменился с указанной даты/имеет другую ETag".

**Ответ сервера**: если ресурс не изменился, сервер отвечает статусом 304 Not Modified и пустым телом. Браузер использует кэшированную копию. Если ресурс изменился, сервер отвечает статусом 200 OK и отправляет новую версию файла.

Управляется этот процесс такими заголовками, как Cache-Control, Expires, Last-Modified и ETag.

1. Что такое сессия в контексте HTTP и как она реализуется?

Сессия - это механизм для поддержания "состояния" (state) между несколькими HTTP-запросами от одного и того же пользователя, несмотря на то, что сам протокол HTTP не имеет состояния (stateless). Сервер "забывает" клиента после каждого запроса. Чтобы реализовать логины, корзины покупок и т.д., нужно как-то связать отдельные запросы в одну логическую сессию.

Реализация:

* Создание сессии: При первом обращении пользователя (например, при логине) сервер создает уникальный идентификатор сессии (Session ID) и сохраняет его в своей памяти или базе данных вместе с данными пользователя (логин, содержимое корзины).
* Передача Session ID: Сервер отправляет этот Session ID клиенту через заголовок ответа Set-Cookie.
* Подтверждение клиентом: Браузер автоматически сохраняет эту "куку" (cookie) и включает ее в заголовок Cookie каждого последующего запроса к этому домену.
* Идентификация: Сервер, получая запрос с заголовком Cookie, извлекает из него Session ID, находит в своей памяти соответствующие данные сессии и "вспоминает" пользователя.

1. Как использовать Wireshark для захвата HTTP-трафика?

Процесс захвата HTTP-трафика в Wireshark состоит из следующих шагов:

Загрузка и установка: Wireshark и библиотека для захвата пакетов (Npcap для Windows).

Выбор сетевого интерфейса:

На главном экране выбрать из списка доступных сетевых подключений (Ethernet, Wi-Fi и т.д.).

Начало захвата: Щелкнуть по выбранному интерфейсу или выбрать в меню Захват (Capture) -> Начать (Start). Можно использовать горячую клавишу Ctrl+E. Wireshark начнет в реальном времени перехватывать все сетевые пакеты, проходящие через выбранный интерфейс.

Генерация трафика: Перейдите на любой веб-сайт по HTTP (не HTTPS), чтобы сгенерировать HTTP-запросы, которые Wireshark сможет перехватить.

Остановка захвата: После того как нужный трафик был перехвачен, выбрать в меню Захват (Capture) -> Остановить (Stop) или снова нажмите Ctrl+E.

1. Что такое фильтры в Wireshark и как они помогают в анализе трафика?

Фильтры в Wireshark — это правила, которые позволяют отображать или захватывать только те сетевые пакеты, которые соответствуют заданным критериям.

Фильтры помогают: сократить "шум": сосредоточиться только на нужном трафике (например, только HTTP); выделить проблему: найти пакеты, связанные с конкретным IP-адресом, портом или ошибкой; провести целевой анализ: изучить поведение конкретного протокола.

1. Как можно просмотреть содержимое HTTP-запроса и ответа в Wireshark?

* Применить фильтр http для отображения только HTTP-пакетов.
* Выбрать нужный пакет в верхней панели списка пакетов (например, с методом GET или кодом ответа 200).
* Анализировать детали в средней панели: Развернуть дерево пункта Hypertext Transfer Protocol.
* Здесь отображаются: метод запроса, URI, версия протокола, все заголовки (Host, User-Agent и т.д.) и код состояния ответа.
* Просмотреть тело сообщения (если есть) можно в нижней панели (в шестнадцатеричном и текстовом представлении) или развернув соответствующий пункт в средней панели (например, Line-based text data).

1. Что такое TCP и как он взаимодействует с HTTP?

TCP (Transmission Control Protocol) — это транспортный протокол, обеспечивающий надежную, упорядоченную и проверенную на ошибки доставку данных между приложениями.

HTTP, как протокол прикладного уровня, использует TCP как транспортный механизм. Перед обменом HTTP-сообщениями клиент и сервер должны установить TCP-соединение (т.н. "тройное рукопожатие"). Все HTTP-запросы и ответы разбиваются TCP на сегменты, передаются по сети и собираются в правильном порядке на принимающей стороне. После завершения обмена HTTP-сообщениями TCP-соединение может быть закрыто.

1. Каковы особенности работы HTTP/2 по сравнению с HTTP/1.1?

* Бинарная кодировка, вместо текстовых команд HTTP/2 использует бинарные кадры, что ускоряет разбор и уменьшает ошибки.
* Мультиплексирование, позволяет передавать несколько потоков данных (запросов и ответов) одновременно в рамках одного TCP-соединения, устраняя проблему блокировки в HTTP/1.1.
* Сжатие заголовков HPACK, заголовки сжимаются, что значительно снижает накладные расходы.
* Server Push, сервер может самостоятельно отправить ресурсы клиенту, которые, по его прогнозу, понадобятся позже.

1. Что такое HTTP-заголовок Referer и для чего он используется?

Заголовок Referer (с орфографической ошибкой, которая стала стандартом) содержит адрес (URI) веб-страницы, с которой был осуществлен переход на текущий запрашиваемый ресурс. Он используется для: понимания источников трафика на сайте, отслеживания того, как пользователи перемещаются по сайту и может использоваться для определения стратегии кэширования.

1. Каковы основные принципы работы с куками в HTTP?

Куки (cookies) — это механизм для сохранения состояния на стороне клиента.

* Создание: Сервер отправляет в ответе заголовок Set-Cookie с данными.
* Хранение: Браузер клиента сохраняет эту пару "имя-значение".
* Отправка: При последующих запросах к тому же домену браузер автоматически включает эти данные в заголовок Cookie.
* Использование: Сервер читает заголовок Cookie и "вспоминает" состояние клиента (например, логин пользователя или содержимое корзины).

1. Что такое Content-Type и как он влияет на обработку данных?

Заголовок Content-Type указывает MIME-тип данных, содержащихся в теле HTTP-сообщения. Он сообщает клиенту или серверу, как интерпретировать переданные данные.

1. Как можно использовать Wireshark для анализа проблем с производительностью HTTP?

* Фильтрация по медленным ответам: Использование фильтра http.time > 0.5 для поиска HTTP-транзакций, занявших больше 500 мс.
* Анализ "водопадной" диаграммы: Просмотр статистики Statistics > HTTP > Packet Counter или использование инструментов в Statistics > Conversations для выявления задержек между запросами.
* Поиск ошибок: Фильтрация по кодам состояния http.response.code >= 400 для нахождения клиентских и серверных ошибок.
* Анализ TCP-потока: Просмотр TCP-потока, связанного с HTTP-запросом, позволяет выявить проблемы с установлением соединения (TCP handshake) или чрезмерную повторную передачу пакетов (TCP retransmissions), что напрямую влияет на скорость HTTP.

1. Что такое REST и как он связан с протоколом HTTP?

REST (Representational State Transfer) — это архитектурный стиль проектирования веб-сервисов (API), а не протокол.

REST использует HTTP как протокол передачи, опираясь на его основные принципы и методы.