КАЗАНСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ
Институт вычислительной математики и информационных технологий
Кафедра системного анализа и информационных технологий

Реферат
по дисциплине «Компьютерные сети»

«Протоколы HTTP и HTTPS»

Выполнил: студент группы 09-032
Каримов Камиль

Казань – 2022

Содержание

HTTP

HTTP (HyperText Transfer Protocol — протокол передачи гипертекста) — символьно-ориентированный клиент-серверный протокол прикладного уровня без сохранения состояния, используемый сервисом World Wide Web.

Основным объектом манипуляции в HTTP является ресурс, на который указывает URI (Uniform Resource Identifier – уникальный идентификатор ресурса) в запросе клиента. Основными ресурсами являются хранящиеся на сервере файлы, но ими могут быть и другие логические (напр. каталог на сервере) или абстрактные объекты (напр. ISBN). Протокол HTTP позволяет указать способ представления (кодирования) одного и того же ресурса по различным параметрам: mime-типу, языку и т. д. Благодаря этой возможности клиент и веб-сервер могут обмениваться двоичными данными, хотя данный протокол является текстовым.

Основные свойства

Основой HTTP является технология «клиент-сервер», то есть предполагается существование:

- Потребителей (клиентов), которые инициируют соединение и посылают запрос;
- Поставщиков (серверов), которые ожидают соединения для получения запроса, производят необходимые действия и возвращают обратно сообщение с результатом.

HTTP в настоящее время повсеместно используется во Всемирной паутине для получения информации с веб-сайтов. В 2006 году в Северной Америке доля HTTP-трафика превысила долю P2P-сетей и составила 46 %, из которых почти половина — это передача потокового видео и звука.

HTTP — протокол прикладного уровня; аналогичными ему являются FTP и SMTP. Обмен сообщениями идёт по обыкновенной схеме «запрос-ответ». Для идентификации ресурсов HTTP использует глобальные URI. В отличие от многих других протоколов, HTTP не сохраняет своего состояния. Это означает отсутствие сохранения промежуточного состояния между парами «запрос-ответ». Компоненты, использующие HTTP, могут самостоятельно осуществлять сохранение информации о состоянии, связанной с последними запросами и ответами (например, «куки» на стороне клиента, «сессии» на стороне сервера). Браузер, посылающий запросы, может отслеживать задержки ответов. Сервер может хранить IP-адреса и заголовки запросов последних клиентов. Однако сам протокол не осведомлён о предыдущих запросах и ответах, в нём не предусмотрена внутренняя поддержка состояния, к нему не предъявляются такие требования.

Большинство протоколов предусматривает установление TCP-сессии, в ходе которой один раз происходит авторизация, и дальнейшие действия выполняются в контексте этой авторизации. HTTP же устанавливает отдельную TCP-сессию на

каждый запрос; в более поздних версиях HTTP было разрешено делать несколько запросов в ходе одной TCP-сессии, но браузеры обычно запрашивают только страницу и включённые в неё объекты (картинки, каскадные стили и т. п.), а затем сразу разрывают TCP-сессию. Для поддержки авторизованного (неанонимного) доступа в HTTP используются cookies; причём такой способ авторизации позволяет сохранить сессию даже после перезагрузки клиента и сервера.

При доступе к данным по FTP или по файловым протоколам тип файла (точнее, тип содержащихся в нём данных) определяется по расширению имени файла, что не всегда удобно. HTTP перед тем, как передать сами данные, передаёт заголовок «Content-Type: тип/подтип», позволяющий клиенту однозначно определить, каким образом обрабатывать присланные данные.

Перечисленные особенности HTTP позволили создавать поисковые машины (первой из которых стала AltaVista, созданная фирмой DEC), форумы и Internet-магазины. Это коммерциализировало Интернет, появились компании, основным полем деятельности которых стало предоставление доступа в Интернет (провайдеры) и создание сайтов.

История развития

Создателем протокола HTTP является Тим Бернерс-Ли, работавший в то время со своей командой в исследовательской организации ЦЕРН (CERN) в 1989 году. Изначально HTTP был задуман как способ реализации сети взаимосвязанных компьютеров, целью создания которой являлась возможность обеспечить доступ к исследованиям и связать их. Предполагалось, что компьютеры в этой сети смогут легко ссылаться друг на друга в режиме реального времени (достаточно будет кликнуть по ссылке – и откроется связанный документ). Идея создания такой системы зародилась достаточно давно, а термин «гипертекст» появился еще в 1960-х. 1980-е характеризуются бурным ростом и развитием Internet. Неудивительно, что именно в то время и появилась возможность реализовать эту идею. В 1989 и 1990 годах Бернерс-Ли опубликовал предложение о создании такой системы. Кроме того, он создал первый веб-сервер на основе HTTP и первый веб-браузер, который мог запрашивать HTML-документы и отображать их.

- **HTTP/0.9**

  HTTP был предложен в марте 1991 года Тимом Бернерсом-Ли, работавшим тогда в CERN, как механизм для доступа к документам в Интернете и облегчения навигации посредством использования гипертекста. Самая ранняя версия протокола HTTP/0.9 была впервые опубликована в январе 1992 года (хотя реализация датируется 1990 годом). Спецификация протокола привела к упорядочению правил взаимодействия между клиентами и серверами HTTP, а также чёткому разделению функций между этими двумя компонентами. Были задокументированы основные синтаксические и семантические положения.

- **HTTP/1.0**

  В мае 1996 года для практической реализации HTTP был выпущен информационный документ RFC 1945, что послужило основой для реализации большинства компонентов HTTP/1.0.

- **HTTP/1.1**

  Современная версия протокола; принята в июне 1999 года[4]. Новым в этой версии был режим «постоянного соединения»: TCP-соединение может оставаться открытым после отправки ответа на запрос, что позволяет посылать несколько запросов за одно соединение. Клиент теперь обязан посылать информацию об имени хоста, к которому он обращается, что сделало возможной более простую организацию виртуального хостинга.

- **HTTP/2**

  11 февраля 2015 года опубликованы финальные версии черновика следующей версии протокола. В отличие от предыдущих версий, протокол HTTP/2 является бинарным. Среди ключевых особенностей: мультиплексирование запросов, расстановка приоритетов для запросов, сжатие заголовков, загрузка нескольких элементов параллельно посредством одного TCP-соединения, поддержка проактивных push-уведомлений со стороны сервера.

- **HTTP/3**

  HTTP/3 — предлагаемый последователь HTTP/2, который уже используется в Веб на основе UDP вместо TCP в качестве транспортного протокола. Как и HTTP/2, он не объявляет устаревшими предыдущие основные версии протокола. Поддержка HTTP/3 была добавлена в Cloudflare и Google Chrome в сентябре 2019 года и может быть включена в стабильных версиях Chrome и Firefox.

Источник: HTTP/2 в действии Барри Поллард

Программное обеспечение

Всё программное обеспечение для работы с протоколом HTTP разделяется на три большие категории:

- Серверы как основные поставщики услуг хранения и обработки информации (обработка запросов);
- Клиенты — конечные потребители услуг сервера (отправка запроса);
- Прокси (посредники) для выполнения транспортных служб.

Для отличия конечных серверов от прокси в официальной документации используется термин «исходный сервер». Один и тот же программный продукт может одновременно выполнять функции клиента, сервера или посредника в зависимости от поставленных задач. В спецификациях протокола HTTP подробно описывается поведение для каждой из этих ролей.

- **Клиенты**

Первоначально протокол HTTP разрабатывался для доступа к гипертекстовым документам Всемирной паутины. Поэтому основными реализациями клиентов являются браузеры (агенты пользователя). Для просмотра сохранённого содержимого сайтов на компьютере без соединения с Интернетом были придуманы офлайн-браузеры. При нестабильном соединении для загрузки больших файлов используются менеджеры закачек; они позволяют в любое время докачать указанные файлы после потери соединения с веб-сервером. Некоторые виртуальные атласы (такие как Google Планета Земля и NASA World Wind) тоже используют HTTP.

Нередко протокол HTTP используется программами для скачивания обновлений.

Целый комплекс программ-роботов используется в поисковых системах Интернета. Среди них веб-пауки (краулеры), которые производят проход по гиперссылкам, составляют базу данных ресурсов серверов и сохраняют их содержимое для дальнейшего анализа.

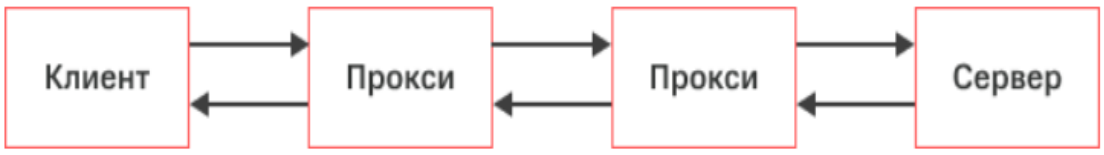
- **Исходные серверы**

Запрос от клиента в конечном итоге приходит на веб-сервер. Он, в свою очередь, отдает документ по запросу клиента. Кстати, стоит помнить, что роль веб-сервера может играть и одна виртуальная машина (*ВМ*), и сразу несколько, которые делят между собой нагрузку и по очереди отвечают на запросы.

Основные реализации: Apache, Internet Information Services (IIS), nginx, LiteSpeed Web Server (LSWS), Google Web Server, lighttpd.

- **Прокси-серверы**

Промежуточный сервер (комплекс программ) в компьютерных сетях, выполняющий роль посредника между пользователем и целевым сервером (при этом о посредничестве могут как знать, так и не знать обе стороны), позволяющий клиентам как выполнять косвенные запросы (принимая и передавая их через прокси-сервер) к другим сетевым службам, так и получать ответы. Сначала клиент подключается к прокси-серверу и запрашивает какой-либо ресурс (например e-mail), расположенный на другом сервере. Затем прокси-сервер либо подключается к указанному серверу и получает ресурс у него, либо возвращает ресурс из собственного кэша (в случаях, если прокси имеет свой кэш). В некоторых случаях запрос клиента или ответ сервера может быть изменён прокси-сервером в определённых целях. Прокси-сервер позволяет защищать компьютер клиента от некоторых сетевых атак и помогает сохранять анонимность клиента, но также может использоваться мошенниками для скрытия адреса сайта, уличённого в мошенничестве, изменения содержимого целевого сайта (подмена), а также перехвата запросов самого пользователя.



Основные реализации: Squid, UserGate, Multiproxy, Naviscope, nginx.

Источники: https://ru.wikipedia.org/wiki/HTTP

Структура HTTP-сообщения

Данные между клиентом и сервером в рамках работы протокола передаются с помощью HTTP-сообщений. Они бывают двух видов:

- **Запросы (HTTP Requests)** — сообщения, которые отправляются клиентом на сервер, чтобы вызвать выполнение некоторых действий. Зачастую для получения доступа к определенному ресурсу. Основой запроса является HTTP-заголовок.
- **Ответы (HTTP Responses)** — сообщения, которые сервер отправляет в ответ на клиентский запрос.

Само по себе сообщение представляет собой информацию в текстовом виде, записанную в несколько строчек.

В целом, как запросы HTTP, так и ответы имеют следующую структуру:

1. Стартовая строка (start line) — используется для описания версии используемого протокола и другой информации — вроде запрашиваемого ресурса или кода ответа. Как можно понять из названия, ее содержимое занимает ровно одну строчку.
2. HTTP-заголовки (HTTP Headers) — несколько строчек текста в определенном формате, которые либо уточняют запрос, либо описывают содержимое тела сообщения.
3. Пустая строка, которая сообщает, что все метаданные для конкретного запроса или ответа были отправлены.
4. Опциональное тело сообщения, которое содержит данные, связанные с запросом, либо документ (например HTML-страницу), передаваемый в ответе.

Источник: https://selectel.ru/blog/http-request

Стартовая строка

Стартовые строки различаются для запроса и ответа. Строка запроса выглядит так:

GET URI — для версии протокола 0.9;

Метод URI HTTP/Версия — для остальных версий.

Здесь:

- **Метод** — тип запроса, одно слово заглавными буквами. В версии HTTP 0.9 использовался только метод **GET**, список методов для версии 1.1 показан далее.
- **URI** определяет путь к запрашиваемому документу.
- **Версия**— пара разделённых точкой цифр. Например: 1.0

Пример запроса:

GET / HTTP/1.0

Host: www.google.com

Стартовая строка ответа сервера имеет следующий формат: **HTTP/Версия Код Состояния Пояснение**, где:

- **Версия** — пара разделённых точкой цифр, как в запросе;
- **Код** состояния — три цифры. По коду состояния определяется дальнейшее содержимое сообщения и поведение клиента;
- **Пояснение** — текстовое короткое пояснение к коду ответа для пользователя. Никак не влияет на сообщение и является необязательным.

Например, стартовая строка ответа сервера может выглядеть так:

HTTP/1.0 200 OK

Источник: https://ru.wikipedia.org/wiki/HTTP

Методы

Методы позволяют указать конкретное действие, которое мы хотим, чтобы сервер выполнил, получив наш запрос. Так, некоторые методы позволяют браузеру (который в большинстве случаев является источником запросов от клиента) отправлять дополнительную информацию в теле запроса — например, заполненную форму или документ.

GET - Позволяет запросить некоторый конкретный ресурс. Дополнительные данные могут быть переданы через строку запроса (Query String) в составе URL (например ?param=value).

POST - Позволяет отправить данные на сервер. Поддерживает отправку различных типов файлов, среди которых текст, PDF-документы и другие типы данных в двоичном виде. Обычно метод POST используется при отправке информации (например, заполненной формы логина) и загрузке данных на веб-сайт, таких как изображения и документы.

HEAD - обычно сервер в ответ на запрос возвращает заголовок и тело, в котором содержится запрашиваемый ресурс. Данный метод при использовании его в запросе позволит получить только заголовки, которые сервер бы вернул при получении GET-запроса к тому же ресурсу. Запрос с использованием данного метода обычно производится для того, чтобы узнать размер запрашиваемого ресурса перед его загрузкой.

PUT - Используется для создания (размещения) новых ресурсов на сервере. Если на сервере данный метод разрешен без надлежащего контроля, то это может привести к серьезным проблемам безопасности.

DELETE - Позволяет удалить существующие ресурсы на сервере. Если использование данного метода настроено некорректно, то это может привести к атаке типа «Отказ в обслуживании» (Denial of Service, DoS) из-за удаления критически важных файлов сервера.

OPTIONS - Позволяет запросить информацию о сервере, в том числе информацию о допускаемых к использованию на сервере HTTP-методов.

PUTCH - Позволяет внести частичные изменения в указанный ресурс по указанному расположению.

Источник: https://selectel.ru/blog/http-request

Коды состояния

Код состояния информирует клиента о результатах выполнения запроса и определяет его дальнейшее поведение.

Каждый код представляется целым трехзначным числом. Первая цифра указывает на класс состояния, последующие - порядковый номер состояния. За кодом ответа обычно следует краткое описание на английском языке.

1xx Informational (Информационный) - В этот класс выделены коды, информирующие о процессе передачи. В HTTP/1.0 сообщения с такими кодами должны игнорироваться. В HTTP/1.1 клиент должен быть готов принять этот класс сообщений как обычный ответ, но ничего отправлять серверу не нужно. Сами сообщения от сервера содержат только стартовую строку ответа и, если требуется, несколько специфичных для ответа полей заголовка. Прокси-сервера подобные сообщения должны отправлять дальше от сервера к клиенту.

Примеры ответов сервера:

- 100 Continue (Продолжать)
- 101 Switching Protocols (Переключение протоколов)
- 102 Processing (Идёт обработка)

2xx Success (Успешно) - Коды состояния из этой категории возвращаются в случае успешной обработки клиентского запроса.

Примеры ответов сервера:

- 200 OK (Успешно).
- 201 Created (Создано)
- 202 Accepted (Принято)

- 204 No Content (Нет содержимого)
- 206 Partial Content (Частичное содержимое)

3xx Redirection (Перенаправление) - Эта категория содержит коды, которые возвращаются, если серверу нужно перенаправить клиента.

Примеры ответов сервера:

- 300 Multiple Choices (Множественный выбор)
- 301 Moved Permanently (Перемещено навсегда)
- 304 Not Modified (Не изменялось)

4xx Client Error (Ошибка клиента) - Коды данной категории означают, что на стороне клиента был отправлен некорректный запрос. Например, клиент в запросе указал не поддерживаемый метод или обратился к ресурсу, к которому у него нет доступа.

Примеры ответов сервера:

- 401 Unauthorized (Неавторизован)
- 402 Payment Required (Требуется оплата)
- 403 Forbidden (Запрещено)
- 404 Not Found (Не найдено)
- 405 Method Not Allowed (Метод не поддерживается)

5xx Server Error (Ошибка сервера) - Ответ с кодами из этой категории приходит, если на стороне сервера возникла ошибка.

Примеры ответов сервера:

- 500 Internal Server Error (Внутренняя ошибка сервера)
- 502 Bad Gateway (Плохой шлюз)
- 503 Service Unavailable (Сервис недоступен)
- 504 Gateway Timeout (Шлюз не отвечает)

Источник: https://selectel.ru/blog/http-request

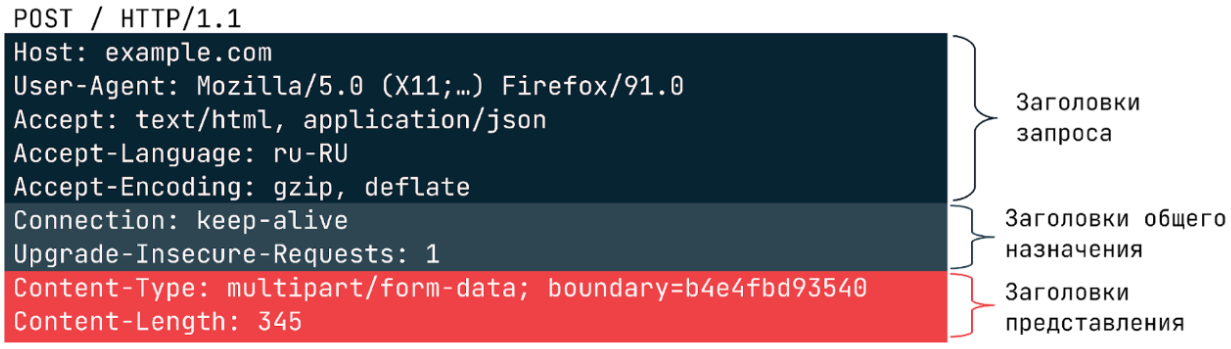
Заголовки

HTTP-заголовок представляет собой строку формата «Имя-Заголовок:Значение», с двоеточием(:) в качестве разделителя. Название заголовка не учитывает регистр, то есть между Host и host, с точки зрения HTTP, нет никакой разницы. Однако в названиях заголовков принято начинать каждое новое слово с заглавной буквы. Структура значения зависит от конкретного заголовка. Несмотря на то, что заголовок вместе со значениями может быть достаточно длинным, занимает он всего одну строчку.

В запросах может передаваться большое число различных заголовков, но все их можно разделить на три категории:

1. **Общего назначения**, которые применяются ко всему сообщению целиком.
2. **Заголовки запроса** уточняют некоторую информацию о запросе, сообщая дополнительный контекст или ограничивая его некоторыми логическими условиями.
3. **Заголовки представления**, которые описывают формат данных сообщения и используемую кодировку. Добавляются к запросу только в тех случаях, когда с ним передается некоторое тело.

Ниже можно видеть пример заголовков в запросе:



Самые частые заголовки запроса:

Host - Используется для указания того, с какого конкретно хоста запрашивается ресурс. В качестве возможных значений могут использоваться как доменные имена, так и IP-адреса. На одном HTTP-сервере может быть размещено несколько различных веб-сайтов. Для обращения к какому-то конкретному требуется данный заголовок

User-Agent - Заголовок используется для описания клиента, который запрашивает ресурс. Он содержит достаточно много информации о пользовательском окружении. Например, может указать, какой браузер используется в качестве клиента, его версию, а также операционную систему, на которой этот клиент работает.

Refer - Используется для указания того, откуда поступил текущий запрос. Например, если вы решите перейти по какой-нибудь ссылке из какого-нибудь сайта, то вероятнее всего к запросу будет добавлен заголовок Refer: название сайта

Accept - Позволяет указать, какой тип медиафайлов принимает клиент. В данном заголовке могут быть указаны несколько типов, перечисленные через запятую (‘ , ‘). А для указания того, что клиент принимает любые типы, используется следующая последовательность — */*.

Cookie - Данный заголовок может содержать в себе одну или несколько пар «Куки-Значение» в формате cookie=value. Куки представляют собой небольшие фрагменты данных, которые хранятся как на стороне клиента, так и на сервере, и выступают в качестве идентификатора. Куки передаются вместе с запросом для поддержания доступа клиента к ресурсу. Помимо этого, куки могут использоваться и для других целей, таких как хранение пользовательских предпочтений на сайте и отслеживание клиентской сессии. Несколько кук в одном заголовке могут быть перечислены с помощью символа точка с запятой (‘ ; ‘), который используется как разделитель.

Authorization - Используется в качестве еще одного метода идентификации клиента на сервере. После успешной идентификации сервер возвращает токен, уникальный для каждого конкретного клиента. В отличие от куки, данный токен хранится исключительно на стороне клиента и отправляется клиентом только по запросу сервера. Существует несколько типов аутентификации, конкретный метод определяется тем веб-сервером или веб-приложением, к которому клиент обращается за ресурсом.

Источник: https://selectel.ru/blog/http-request

Тело запроса

Завершающая часть HTTP-запроса — это его тело. Не у каждого HTTP-метода предполагается наличие тела. Так, например, методам вроде GET, HEAD, DELETE, OPTIONS обычно не требуется тело. Некоторые виды запросов могут отправлять данные на сервер в теле запроса: самый распространенный из таких методов — POST.

Пример HTTP-диалога

Запрос клиента:

GET /wiki/страница HTTP/1.1

Host: ru.wikipedia.org

User-Agent: Mozilla/5.0 (X11; U; Linux i686; ru; rv:1.9b5) Gecko/2008050509 Firefox/3.0b5

Accept: text/html

Connection: close

(пустая строка)

Ответ сервера:

HTTP/1.1 200 OK

Date: Wed, 11 Feb 2009 11:20:59 GMT

Server: Apache

X-Powered-By: PHP/5.2.4-2ubuntu5wm1

Last-Modified: Wed, 11 Feb 2009 11:20:59 GMT

Content-Language: ru

Content-Type: text/html; charset=utf-8

Content-Length: 1234

Connection: close

(пустая строка)

(запрошенная страница в HTML)

Источник: https://ru.wikipedia.org/wiki/HTTP

HTTPS

HTTPS (от англ. HyperText Transfer Protocol Secure) – это безопасный протокол передачи данных, который поддерживает шифрование посредством криптографических протоколов SSL и TLS, и является расширенной версией протокола HTTP.

Несмотря на свою функциональность у HTTP есть один очень важный недостаток — **незащищённость**. Данные между пользователями передаются в открытом виде, злоумышленник может вмешаться в передачу данных, перехватить их или изменить. Чтобы защитить данные пользователей, был создан протокол **HTTPS**.

HTTPS работает благодаря SSL/TLS-сертификату. **SSL/TLS-сертификат —** это цифровая подпись сайта. С её помощью подтверждается его подлинность. Перед тем как установить защищённое соединение, браузер запрашивает этот документ и обращается к центру сертификации, чтобы подтвердить легальность документа. Если он действителен, то браузер считает этот сайт безопасным и начинает обмен данными.

Система HTTPS похожа на провод, который состоит из двух слоёв: медная сердцевина и оболочка. Медная сердцевина — основная часть провода, по которой идёт ток. Оболочка защищает контакты от внешних воздействий. Так, медная сердцевина — это HTTP-протокол, а защитная оболочка — это SSL-сертификат. Такое сотрудничество создаёт безопасное HTTPS-соединение.

Источник: https://ru.wikipedia.org/wiki/HTTPS

Ключи шифрования

Кроме подтверждения подлинности сайта, SSL-сертификат шифрует данные. После того как браузер убедился в подлинности сайта, начинается обмен шифрами. Шифрование HTTPS происходит при помощи симметричного и асимметричного ключа. Вот что это значит:

- **Асимметричный** ключ — каждая сторона имеет два ключа: публичный и частный. Публичный ключ доступен любому. Частный известен только владельцу. Если браузер хочет отправить сообщение, то он находит публичный ключ сервера, шифрует сообщение и отправляет на сервер. Далее сервер расшифровывает полученное сообщение с помощью своего частного ключа. Чтобы ответить пользователю, сервер делает те же самые действия: поиск публичного ключа собеседника, шифрование, отправка.
- **Симметричный** ключ — у обеих сторон есть один ключ, с помощью которого они передают данные. Между двумя сторонами уже должен быть установлен первичный контакт, чтобы браузер и сервер знали, на каком языке общаться.

Чтобы установить HTTPS-соединение, браузеру и серверу надо договориться о симметричном ключе. Для этого сначала браузер и сервер обмениваются асимметрично зашифрованными сообщениями, где указывают секретный ключ и далее общаются при помощи симметричного шифрования.

Итак, какова функция протокола HTTPS?

1. **Шифрование**. Информация передаётся в зашифрованном виде. Благодаря этому злоумышленники не могут украсть информацию, которой обмениваются посетители сайта, а также отследить их действия на других страницах.

2. **Аутентификация**. Посетители уверены, что переходят на официальный сайт компании, а не на дубликат, сделанный злоумышленником.
3. **Сохранение** данных. Протокол фиксирует все изменения данных. Если злоумышленник всё-таки пытался взломать защиту, об этом можно узнать из сохранённых данных.

Также стоит упомянуть, какой порт используется протоколом HTTPS по умолчанию. HTTPS использует для подключения 443 порт — его не нужно дополнительно настраивать

Принцип работы

Итак, протокол HTTPS предназначен для безопасного соединения. Чтобы понять, как устанавливается это соединение и как работает HTTPS, рассмотрим механизм пошагово.

Для примера возьмём ситуацию: пользователь хочет перейти на сайт google.com, который работает по безопасному протоколу HTTPS.

1. Браузер пользователя просит предоставить SSL-сертификат.
2. Сайт на HTTPS отправляет сертификат.
3. Браузер проверяет подлинность сертификата в центре сертификации.
4. Браузер и сайт договариваются о симметричном ключе при помощи асимметричного шифрования.
5. Браузер и сайт передают зашифрованную информацию.

Заключение

Internet стал неотъемлемой частью повседневной жизни. В нем мы совершаем покупки, производим банковские операции, а также общаемся и развлекаемся. По мере развития интернета вещей (IoT – Internet of Things) к сети подключается все больше и больше устройств, что обеспечивает нам возможность получения удаленного доступа к ним. Реализация такого доступа стала возможной благодаря разработке нескольких технологий, в числе которых протокол передачи гипертекста (Hypertext Transfer Protocol, HTTP). Поэтому можно однозначно сказать, что эта технология занимает важное место в компьютерных сетях.

Список источников

– Печатные издания

HTTP/2 в действии Барри Поллард

– Электронные ресурсы

https://ru.wikipedia.org/wiki/HTTP

https://selectel.ru/blog/http-request

https://ru.wikipedia.org/wiki/HTTPS