Лабораторная работа № 7

Анализ и исследование протокола прикладного уровня HTTP и его сообщений

Цель работы

Анализ и исследование протокола HTTP путем имитации запросов HTTP клиента к серверу с помощью утилиты Telnet.

Теоретические сведения

HTTP (Hypertext Transfer Protocol)

HTTP — протокол прикладного уровня, разработанный для обмена гипертекстовой информацией в сети Internet. Протокол используется одной из популярнейших систем Сети — Word Wide Web — с 1990 года.

HTTP позволяет реализовать в рамках обмена данными набор методов, базирующихся на спецификации универсального идентификатора ресурсов (Universal Resource Identifier — URI). HTTP Протокол реализует принцип «запрос/ответ». Запрашивающая программа — клиент — инициирует взаимодействие с отвечающей программой — сервером, и посылает запрос, включающий в себя HTTP метод, URI ресурса, версию протокола, сообщение с модификаторами типа передаваемой информации, информацию клиента, и, возможно, тело запроса. Сервер отвечает строкой состояния, включающей версию протокола и код состояния, за которой следует сообщение. Данное сообщение содержит информацию сервера, метаинформацию и тело ответа.

При работе в Internet для обслуживания HTTP запросов как правило используется 80 порт TCP. Общий сценарий работы таков: клиент устанавливает соединение и отсылает запрос. После отправки ответа сервер, в зависимости от модификаторов запроса, либо инициирует разрыв соединения, либо ожидает следующего запроса от клиента.

Ниже рассматривается версия 1.1 протокола HTTP.

Запрос клиента может выглядеть, например, так:

```
HEAD / HTTP/1.1
Host: www.w3.org
```

В этой записи слово «GET» обозначает HTTP метод GET; «www.w3.org» — имя узла, на котором находится ресурс; «/» — относительный URI ресурса.

Ответ сервера, может выглядеть следующим образом:

```
HTTP/1.1 200 OK
Date: Sat, 22 May 2004 13:55:57 GMT
Server: Apache/1.3.28 (Unix) PHP/4.2.3
Content-Location: Overview.html
Vary: negotiate,accept
TCN: choice
P3P: policyref="http://www.w3.org/2001/05/P3P/p3p.xml"
Cache-Control: max-age=600
Expires: Sat, 22 May 2004 14:05:57 GMT
Last-Modified: Mon, 17 May 2004 13:25:50 GMT
ETag: "40a8bd5e;3e2eee38"
Accept-Ranges: bytes
Content-Length: 22342
Content-Type: text/html; charset=utf-8
```

Здесь первая строка — указание статуса ответа. Далее идут поля заголовка, содержащие информацию, характеризующую обстоятельства ответа и запрашиваемый ресурс.

URI (Universal Resource Identifier — унифицированый идентификатор ресурса)

URI появился, когда Internet стал по-настоящему глобальным как по территориальному критерию, так и по типу ресурсов и сервисов, которые стали доступны с его помощью. Возникла потребность в способе уникально идентифицировать любой ресурс Сети и способ доступа к нему. В результате была создана концепция URI.

В упрощенном общем случае вид URI таков:

схема://узел:порт/путь?запрос

Обязательной здесь является только схема, наличие же или отсутствие остальных элементов определяется этой схемой. Схема определяет формат URI, а также метод доступа к ресурсу.

В большинстве схем явное указание номера порта необязательно, так как эти схемы определяют доступ по определенному TCP протоколу, для которого существует общепринятый номер порта. Так, например, для HTTP обычно используется порт 80.

И в литературе, и в устном общении часто употребляются термины URL (Uniform Resource Locator) и URN (Uniform Resource Name). В RFC 2396 рекомендовано использовать вместо них единый термин URI.

Наиболее употребимые схемы URI

- file — локальный файл.
- ftp — FTP ресурс.
- http — HTTP ресурс.
- mailto — адрес электронной почты.
- telnet — сервер Telnet.

Примеры URI

- http://www.w3.org/Overview.html — HTML файл, доступный по протоколу HTTP.
- http://www.google.com/custom?q=search_string&domains=www.w3.org — данные, доступные по протоколу HTTP, с указанием дополнительных параметров ресурса.
- file://\D:\Doc\HTML4\intro\intro.html — локальный файл Microsoft Windows, содержащий HTML текст.
- ftp://ftp.w3.org/welcome.msg — текстовый файл, доступный по протоколу FTP.
- mailto:mduerst@ifi.unizh.ch — адрес электронной почты mduerst@ifi.unizh.ch.
- telnet://melvyl.ucop.edu/ — Telnet сервер melvyl.ucop.edu.

Относительные URI

Относительные URI отличаются тем, что содержат только запрос и/или путь. Они применяются в тех случаях, когда тем или иным способом определен базовый URI для набора ресурсов, тогда URI этих ресурсов указываются относительно базового URI. Например, в HTTP запросе базовый URI неявно задается с помощью поля запроса «Host». Например, если это поле содержит имя узла www.w3.org, то базовый URI будет выглядеть как http://www.w3.org/.

В HTTP запросах относительные URI позволяют упростить его текст. Например, запрос:

GET http://www.w3.org/ HTTP/1.1
Host: www.w3.org
может быть записан как:
GET / HTTP/1.1

Host: www.w3.org

Методы HTTP

В настоящее время в практике World Wide Web реально используются только три HTTP метода: POST, GET и HEAD.

GET

Позволяет получить данные, определенные с помощью URI в запросе ресурса. Дополнительные данные, которые надо передать для обработки, кодируются в URI. При использовании метода GET в теле ответа возвращается затребованная информация, как правило — текст HTML документа.

HEAD

Метод, аналогичный GET, отличающийся тем, что возвращает только заголовок ответа. Используется для получения информации о ресурсе.

POST

Этот метод используется для передачи большого объема информации на сервер. Как правило, им пользуются для загрузки изображений, файлов с локального диска клиента или больших блоков текста. В отличии от GET и HEAD, в POST передается тело запроса, в котором и содержится передаваемая информация.

Малоиспользуемые методы

В первых версиях протокола было определено множество HTTP методов, которые не нашли должного применения. На практике оказалось, что многие функции, возлагавшиеся на эти методы, можно успешно выполнять с помощью POST.

Изменение числа методов доступа отражает практику использования HTTP. Однако, с исторической и методической точек зрения, первые версии протокола представляют несомненный интерес, особенно раздел, описывающий методы доступа. В версии 1993 года насчитывалось 13 различных методов доступа. Среди этих методов были такие, например, как:

- CHECKOUT — защита от несанкционированного доступа.
- PUT — замена содержания информационного ресурса.
- DELETE — удаление ресурса.
- LINK — создание гипертекстовой ссылки.
- UNLINK — удаление гипертекстовой ссылки.
- SPACEJUMP — переход по координатам.
- SEARCH — поиск.

Из списка видно, что протокол был максимально ориентирован на работу с гипертекстовыми распределенными системами, причем не только с точки зрения потребителя, но и с точки зрения разработчика подобных систем. Однако, во-первых, как показывал опыт, практически не использовались методы доступа, связанные с изменением информации. Это объясняется прежде всего соображениями безопасности. Во-вторых, многие методы были с успехом заменены на функционально аналогичные CGI программы — программы, автоматически генерирующие ресурс, при обращении к определенному URI.

HTTP сообщения

Под сообщениями HTTP понимаются как запросы клиента к серверу, так и ответы сервера клиенту.

И запросы и ответы используют для передачи данных формат, определенный для сообщений электронной почты в RFC 822. Оба типа сообщений состоят из первой строки, некоторого количества полей заголовка, пустой строки и, возможно, тела сообщения.

В общем виде HTTP сообщение можно представить следующим образом:

{первая строка}
{имя поля заголовка}: {значение поля заголовка}

...
{пустая строка}
{тело сообщения}

Обязательными элементами HTTP сообщения являются только первая строка и пустая строка.

Запрос

В общем виде HTTP запрос выглядит следующим образом:

{строка запроса}
{имя поля заголовка}: {значение поля заголовка}
...
{пустая строка}
{тело сообщения}

Строка запроса состоит из имени метода HTTP, URI запрашиваемого ресурса и версии протокола. URI запрашиваемого ресурса может быть относительным. Строка запроса может выглядеть следующим образом:

GET http://www.w3.org/ HTTP/1.1

Необходимо отметить, что не смотртя на то, что HTTP сообщение может не иметь ни одного поля заголовка, запрос обязательно должен содержать поле заголовка «Host». Это требование было определено в последней версии протокола и обусловлено проблемами, которые возникали ранее при ситуации, когда одна программа-сервер обслуживала запросы для нескольких виртуальных узлов.

Ответ

В общем виде HTTP ответ выглядит следующим образом:

{строка состояния}
{имя поля заголовка}: {значение поля заголовка}
...
{пустая строка}
{тело сообщения}

Строка состояния состоит из версии протокола, кода состояния и краткого описания этого кода. Например, она может выглядеть так:

HTTP/1.1 200 OK

Код состояния

Код состояния — это число, состоящее из трех цифр и характеризующее результат попытки сервера понять и выполнить запрос клиента. Первая цифра кода состояния определяет тип состояния. Код состояния может иметь следующий вид:

- **1xx** — уведомляющий: запрос получен, обработка продолжается.
- **2xx** — успешное выполнение: запрашиваемое действие успешно принято, понято и выполнено.
- **3xx** — перенаправление: для завершения запрашиваемого действия, клиент должен произвести дополнительные операции.
- **4xx** — ошибка на стороне клиента: запрос имеет неверный формат или не может быть выполнен из-за данных запроса.
- **5xx** — ошибка на стороне сервера: серверу не удалось обработать потенциально корректный запрос.

Наиболее часто встречающиеся коды состояния таковы:

- **100 (Continue)** — клиент должен продолжать передачу запроса. Уже переданная им часть запроса получена и не была отвергнута сервером.
- **200 (OK)** — запрос успешно обработан.
- **204 (No Content)** — сервер выполнил запрос, но ему нечего возвращать клиенту.
- **301 (Moved Permanently)** — запрошенный ресурс сменил свой URI. Его новый URI указан в поле заголовка ответа «Location».

- **400 (Bad Request)** — запрос не был понят сервером из-за его неверного синтаксиса.
- **401 (Unauthorized)** — запрос требует авторизации, тип которой указан в поле заголовка ответа «WWW-Authenticate».
- **403 (Forbidden)** — сервер понял запрос, но отказался его выполнять.
- **404 (Not Found)** — ресурс, заданный в URI запроса, не найден.
- **406 (Not Acceptable)** — ресурс, заданный в URI запроса, может генерировать только ответы, неприемлемые для клиента.
- **408 (Request Timeout)** — клиент не послал ни одного запроса в течение отведенного ему времени.
- **500 (Internal Server Error)** — запрос не выполнен из-за внутренней ошибки сервера.

Поля заголовка

Ниже перечислены поля заголовка и описания их возможных значений. Приведенный список неполон, однако в нем перечислена большая часть часто использующихся полей.

Основные поля заголовка

Поле	Описание
Cache-Control	Директивы управления механизмами кэширования. Например, «no-cache» — для данного ресурса кэширование использваться не должно.
Connection	Параметры соединения. «Keep-Alive» — после ответа сервер ожидает следующего запроса; «Close» — после ответа сервер завершает соединение.
Date	Дата и время посылки данного сообщения.
Pragma	Используется для задания директив, зависящих от реализации. Например, большинство браузеров поддерживает значение данного поля в ответе сервера «no-cache» и, если оно используется, не производят кэширования ресурса.

Поля заголовка запроса

Поле	Описание
Accept	MIME тип данных, который клиент готов принять от сервера.
Accept-Charset	Кодировки символов, приемлемые в ответе. Например, «iso-8859-5», «unicode-1-1».
Accept-Encoding	Кодировки данных, приемлемые в ответе. Например, «compress, gzip» — клиентом поддерживается сжатие методом gzip.
Accept-Language	Естественные языки, приемлемые в ответе. Например, «ru», «en-us».
Authorization	Имя и пароль пользователя для доступа к ресурсу согласно RFC 2617.
From	Адрес электронной почты администратора клиентской программы.
Host	Задает имя и, возможно, порт узла, на котором расположен запрашиваемый ресурс. Например, «www.w3.org».
Referer	URI ресурса, с которого был получен URI запроса. Как правило, содержит адрес HTML страницы, с которой был произведен переход на запрашиваемую.
User-Agent	Идентификатор клиентской программы. Например, «Mozilla/4.0 (compatible; MSIE 5.5; Windows NT 5.0)».

Поля заголовка ответа

Поле	Описание
Accept-Ranges	Указывает, может ли сервер обрабатывать запросы на часть данных ресурса с указанием диапазона. Если «none» — не может; иначе указываются единицы, в которых должен быть указан диапазон, например, «bytes».
Age	Время в секундах, прошедшее с момента генерации ответа исходным сервером.
ETag	Тег (идентификатор) текущего сообщения. Может использоваться для различения результатов идентичных запросов ресурса.
Location	Содержит URI ресурса, отличный от запрошенного, по которому нужно произвести дозапрос, с тем чтобы получить данные. Используется для переадресации.
Server	Информация о программном обеспечении сервера. Например, «Apache/1.3.28 (Unix) PHP/4.2.3».
WWW-Authenticate	Сообщает, что ресурс требует авторизации, и возвращает схему авторизации согласно RFC 2617.

Поля заголовка сообщения

Поле	Описание
Allow	HTTP методы, поддерживаемые запрошенным ресурсом.
Content-Encoding	Задает способ кодирования тела сообщения. Например, «gzip» — тело собщения сжато методом gzip.
Content-Language	Естественный язык, используемый в теле сообщения.
Content-Length	Размер тела сообщения в байтах.
Content-Location	Содержит реальный URI ресурса, когда он отличен от запрашиваемого.
Content-Range	Определяет, какой именно частью большего фрагментированного сообщения является тело данного сообщения. Например, «bytes 0-499/1234» — тело данного сообщения представляет собой байты большего фрагментированного сообщения с нулевого по 499-ый, при этом размер полного сообщения составляет 1234 байта.
Content-Type	MIME тип сообщения.
Expires	Дата и время, после которых содержание ответа считается устаревшим.
Last-Modified	Дата и время последнего изменения передаваемого ресурса.

MIME — Многоцелевые расширения почты в Internet (Multipurpose Internet Mail Extensions).

Способ кодирования/декодирования электронных писем, в котором производится согласование типов передаваемых документов с помощью заголовков. Однако, типы

MIME используются для указания формата не только сообщений электронной почты, но и сообщений многих других сервисов Internet. MIME типы используются и в HTTP.

Примеры MIME типов:

- **text/plain** — обычный текст
- **text/html** — HTML гипертекст
- **image/jpeg** — избражение в формате JPEG
- **text/html; charset=windows-1251** — HTML гипертекст в кодировке windows-1251

Порядок выполнения лабораторной работы

Задание

Используя утилиту Windows Telnet подключиться к HTTP серверу согласно варианту задания, произвести ряд запросов. Проанализировать ответы сервера, заполнить бланк аналитического отчета.

Задание 1. Анализ HTTP сообщений

1. Запустить Wireshark или Http Analyzer и webбраузер. Перейти на страницу сайта согласно заданию.
2. Обновить страницу (F5). Пронаблюдать совершаемые webбраузером запросы, включая адреса ресурсов, заголовки запросов и ответов на них.
3. Идентифицировать элементы webстраницы, получаемые по запросам в анализаторе.
4. Просмотреть запросы и ответы HTTP в Wireshark или Http Analyzer в текстовом виде.
5. Пронаблюдать установление сеанса TLS (HTTPS) в Wireshark или Http Analyzer и установить на каком уровне протоколов шифруются данные и начиная с какого этапа.
6. Результаты свести в отчет.

Задание 2. Работа с локальным HTTP сервером через telnet.

1. Запустите Windows Telnet. Примечание: Компонент Telnet в Windows 7 по умолчанию отключен. Чтобы включить его, делаем следующее:
   - Зайдите в “Пуск” – “Панель управления” – “Программы и компоненты” – “Включение или отключение компонентов Windows”.
   - В открывшемся окне поставьте флажок в строке “Клиент Telnet” – затем нажмите “ОК”:
2. Установите соединение с удаленным хостом. Пример: open localhost 80.
3. Используя HTTP 1.1 запрос HEAD, получите заголовок страницы с HTTP-сервера. Пример HEAD HTTP://LOCALHOST/PERLCALC.HTML HTTP/1.0 <ENTER><ENTER>
4. Внесите текст заголовка ответа в отчет.
5. Чтобы вернуться в командную строку Windows Telnet, дважды нажмите клавишу «Enter».
6. В командной строке Windows Telnet введите: «logfile telnet.log». С этого момента вся входящая и исходящая информация будет записываться в указанный файл.
7. С помощью HTTP 1.1 запроса GET получите содержимое HTML страницы, размещенной на Web-сервере. GET HTTP://LOCALHOST/PERLCALC.HTML HTTP/1.0 <ENTER><ENTER>
8. После завершения соединения с сервером, закройте Windows Telnet.

Задание 3. Обработка данных

1. Завершите заполнение аналитического отчета, занося в таблицу только те поля заголовка ответа, которые описаны в краткой теории.
2. В файле «telnet.log» удалите HTTP заголовок, оставив только данные.
3. Переименуйте файл, задав ему расширение .html.
4. Открыв с помощью Internet Explorer полученный HTML файл, вы должны увидеть, может быть в искаженном виде, ту же информацию, что и при просмотре в броузере Internet Explorer при посещении HTML-страницы указанного сервера.

Примечание. При заполнении отчета в таблице полей заголовка ответа в поле «Пояснение» следует разъяснять не общее назначение поля, а его значение, полученное в практической части работы.

Задание 4. Работа с удаленным HTTP сервером через Http Analyzer.

1. Запустите Http Analyzer.
2. Установите соединение с удаленным хостом согласно варианту задания. Пример: open localhost 80.
3. Генерируйте различные методы запросов HTTP к серверу: запросы HEAD, GET, POST, PUT, TRACE, MOVE, DELETE, CONNECT, OPTION
4. Получите и проанализируйте ответы. Расшифруйте ответы и дайте пояснения.
5. Результаты и полученные данные сохраните для каждого метода в отчете.

Защита работа

Для защиты данной работы, необходимо:

1. Предоставить HTML файл.
2. Предоставить заполненный аналитический отчет.
3. Уметь ответить на контрольные вопросы.