ЛЕКЦИЯ 2. ВЗАИМОДЕЙСТВИЕ КЛИЕНТ-СЕРВЕР В WWW

КЛИЕНТ-СЕРВЕРНЫЕ ТЕХНОЛОГИИ ИНТЕРНЕТ

- Основой протокол *HTTP* является технология «*клиент-сервер*», то есть предполагается, что:
  - потребитель-клиент инициировав соединение с поставщиком-сервером посылает ему запрос;
  - Поставщик-сервер, получив запрос, производит необходимые действия и возвращает обратно клиенту ответ с результатом.
- *Тонкий клиент* — это компьютер-клиент, который переносит все задачи по обработке информации на сервер. Примером тонкого клиента может служить компьютер с *браузером*, использующийся для работы с *веб-приложениями*.
- *Толстый клиент*, напротив, производит обработку информации независимо от сервера, использует последний в основном лишь для хранения данных.

ПРОТОКОЛ HTTP

- ***HTTP*** (*HyperText Transfer Protocol* - RFC 1945, RFC 2616) — протокол прикладного уровня для передачи гипертекста.
- Центральным объектом в *HTTP* является ресурс, на который указывает *URI* в запросе клиента. Обычно такими ресурсами являются хранящиеся на сервере файлы. Особенностью протокола *HTTP* является возможность указать в запросе и ответе способ представления одного и того же ресурса по различным параметрам: формату, кодировке, языку и т. д. Именно благодаря возможности указания способа кодирования сообщения клиент и сервер могут обмениваться двоичными данными, хотя данный протокол является текстовым.

ПРОТОКОЛ HTTP

- В отличие от многих других протоколов, HTTP является протоколом без памяти. Это означает протокол не осведомлён о предыдущих запросах и ответах.
- Компоненты, использующие HTTP, могут самостоятельно осуществлять сохранение информации о состоянии, связанной с последними запросами и ответами.
  - Клиентское веб-приложение, посылающее запросы, может отслеживать задержки ответов.
  - Сервер может хранить IP-адреса и заголовки запросов последних клиентов.

ПРОТОКОЛ HTTP

- Всё программное обеспечение для работы с протоколом HTTP разделяется на три основные категории:
  - Серверы - поставщики услуг хранения и обработки информации (обработка запросов).
  - Клиенты — конечные потребители услуг сервера (отправка запросов).
  - Прокси-серверы для поддержки работы транспортных служб.

ПРОТОКОЛ HTTP

- Основными *клиентами* являются *браузеры* например: *Internet Explorer*, *Opera*, *Mozilla Firefox*, *Netscape Navigator, Chrome* и др.
- Наиболее известными реализациями *веб-серверов* являются: *Internet Information Services* (IIS), *Apache*, *lighttpd*, *nginx*.
- Наиболее известные реализации *прокси-серверов*: *Squid*, *UserGate*, *Multiproxy*, *Naviscope*.

СТРУКТУРА ПРОТОКОЛА HTTP

- Каждое HTTP-сообщение состоит из трёх частей, которые передаются в указанном порядке:
  - ***Заголовок сообщения***, который начинается со ***строки состояния***, определяющей тип сообщения, и ***полей заголовка***, характеризующих тело сообщения, описывающих параметры передачи и прочие сведения;
  - ***Пустая строка***;
  - ***Тело сообщения*** — непосредственно данные сообщения.
- *Поля заголовка* и *тело* сообщения могут отсутствовать, но *строка состояния* является обязательным элементом, так как указывает на тип запроса/ответа.

СТРУКТУРА ПРОТОКОЛА HTTP

URI, URL, URN

- ***URI*** (*Uniform Resource Identifier*) — единообразный идентификатор ресурса, представляющий собой короткую последовательность символов, идентифицирующую абстрактный или физический ресурс.
- Самые известные примеры *URI* — это *URL* и *URN*.
- ***URL*** (*Uniform Resource Locator*) - это *URI*, который, помимо идентификации ресурса, предоставляет ещё и информацию о местонахождении этого ресурса.
- ***URN*** (*Uniform Resource Name*) — это *URI*, который идентифицирует ресурс в определённом пространстве имён, но, в отличие от *URL*, *URN* не указывает на местонахождение этого ресурса.
- *URI* не указывает на то, как получить ресурс, а только идентифицирует его. Что даёт возможность описывать с помощью *RDF* (*Resource Description Framework*) ресурсы, которые не могут быть получены через Интернет (имена, названия и т.п.)

СТРУКТУРА URL

<схема>://<логин>:<пароль>@<хост>:<порт>/<URL>/<путь>

Где:

- *схема* - схема обращения к ресурсу (обычно сетевой протокол);
- *логин* - имя пользователя, используемое для доступа к ресурсу;
- *пароль* - пароль, ассоциированный с указанным именем пользователя;
- *хост* - полностью прописанное доменное имя хоста в системе *DNS* или *IP-адрес* хоста;
- *порт* - порт хоста для подключения;
- *URL/путь* - уточняющая информация о месте нахождения ресурса.

СТРУКТУРА URL

- Общепринятые схемы (протоколы) URL включают протоколы: *ftp*, *http*, *https*, *telnet*, а также:
  - *gopher* — протокол *Gopher*;
  - *mailto* — адрес электронной почты;
  - *news* — новости *Usenet*;
  - *nntp* — новости *Usenet* через протокол *NNTP*;
  - *irc* — протокол *IRC*;
  - *prospero* — служба каталогов *Prospero Directory Service*;
  - *wais* — база данных системы *WAIS*;
  - *xmpp* — протокол *XMPP* (часть *Jabber*);
  - *file* — имя локального файла;
  - *data* — непосредственные данные (*Data*: *URL*);

ПОРТ TCP/IP

- TCP/IP ***порт*** — целое число от 1 до 65535, позволяющие различным программам, выполняемым на одном хосте, получать данные независимо друг от друга. Каждая программа обрабатывает данные, поступающие на определённый порт («слушает» этот порт).
- Самые распространенные сетевые протоколы имеют стандартные номера портов, хотя в большинстве случаев программа может использовать любой порт.
- Для наиболее распространенных протоколов стандартные номера портов следующие:
  - HTTP: 80
  - FTP: 21 (для команд), 20 (для данных)
  - telnet: 23
  - POP3: 110
  - IMAP: 143
  - SMTP: 25
  - SSH: 22

HTTPS

- HTTPS — расширение протокола *HTTP*, поддерживающее шифрование. Данные, передаваемые по протоколу *HTTP*, «упаковываются» в криптографический протокол *SSL* или *TLS*, тем самым обеспечивается защита этих данных. В отличие от *HTTP*, для *HTTPS* по умолчанию используется TCP-порт 443.
- Чтобы подготовить веб-сервер для обработки *HTTPS* соединений, администратор должен получить и установить в систему сертификат для этого веб-сервера.

SSL и TLS

- *SSL* (Secure Sockets Layer) — криптографический протокол, обеспечивающий безопасную передачу данных по сети Интернет.
- При его использовании создаётся защищённое соединение между клиентом и сервером. *SSL* изначально разработан компанией *Netscape Communications*. Впоследствии на основании протокола *SSL 3.0* был разработан и принят стандарт *RFC*, получивший название TLS.
- Протокол использует шифрование с открытым ключом для подтверждения подлинности передатчика и получателя. Поддерживает надёжность передачи данных за счёт использования корректирующих кодов и безопасных хэш-функций.

SSL и TLS

- На нижнем уровне многоуровневого транспортного протокола (например, TCP) он является протоколом записи и используется для инкапсуляции различных протоколов (например POP3, IMAP, SMTP или HTTP).
- Для каждого инкапсулированного протокола он обеспечивает условия, при которых сервер и клиент могут подтверждать друг другу свою подлинность, выполнять алгоритмы шифрования и производить обмен криптографическими ключами, прежде чем протокол прикладной программы начнет передавать и получать данные.
- Для доступа к веб-страницам, защищённым протоколом SSL, в URL вместо схемы http, как правило, подставляется схема https, указывающая на то, что будет использоваться SSL-соединение. Стандартный TCP-порт для соединения по протоколу https — 443.
- Для работы SSL требуется, чтобы на сервере имелся SSL-сертификат.

МЕТОДЫ АУТЕНТИФИКАЦИИ В WWW

- *Basic* — базовая аутентификация, при которой имя пользователя и пароль передаются в заголовках *http-пакетов*. Пароль при этом не шифруется и присутствует в чистом виде в кодировке *base64*. Для данного типа аутентификации использование *SSL* является обязательным.
- *Digest* — дайджест-аутентификация, при которой пароль пользователя передается в хешированном виде. По уровню конфиденциальности паролей этот тип мало чем отличается от предыдущего, так как атакующему все равно, действительно ли это настоящий пароль или только *хеш* от него: перехватив удостоверение, он все равно получает доступ к конечной точке. Для данного типа аутентификации использование *SSL* является обязательным.

МЕТОДЫ АУТЕНТИФИКАЦИИ В WWW

- *Integrated* — интегрированная аутентификация, при которой клиент и сервер обмениваются сообщениями для выяснения подлинности друг друга с помощью протоколов *NTLM* или *Kerberos*. Этот тип аутентификации защищен от перехвата удостоверений пользователей, поэтому для него не требуется протокол *SSL*. Только при использовании данного типа аутентификации можно работать по схеме *http*, во всех остальных случаях необходимо использовать схему *https*.

ЛИТЕРАТУРА