ETU "LETI"
SAINT PETERSBURG ELECTROTECHNICAL UNIVERSITY

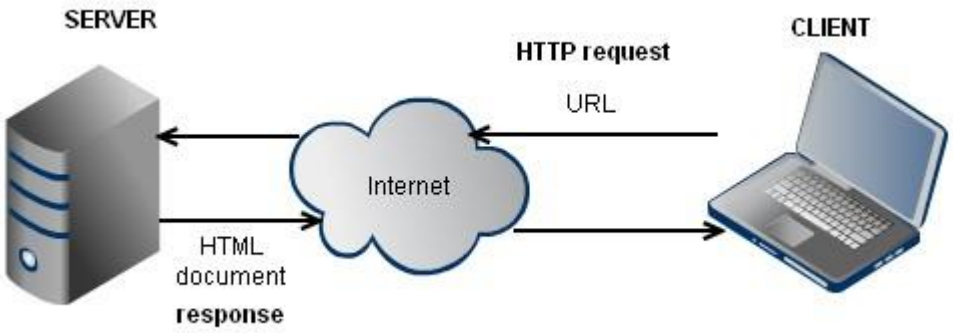
ОСНОВЫ ПОСТРОЕНИЯ ЗАЩИЩЕННЫХ КОМПЬЮТЕРНЫХ СЕТЕЙ

Сетевые протоколы передачи информации.
Протокол HTTP

Архитектура

Как и другие протоколы, отвечающие требованиям модели OSI, HTTP работает на клиент-серверной архитектуре.

- HTTP не сохраняет своего состояния между парами запрос-ответ, но не запрещает сохранять состояние контрагентам.
- HTTP соединение обычно происходит на базе TCP/IP соединений.
- Порт TCP по умолчанию - 80, но могут использоваться и другие порты.



Клиент

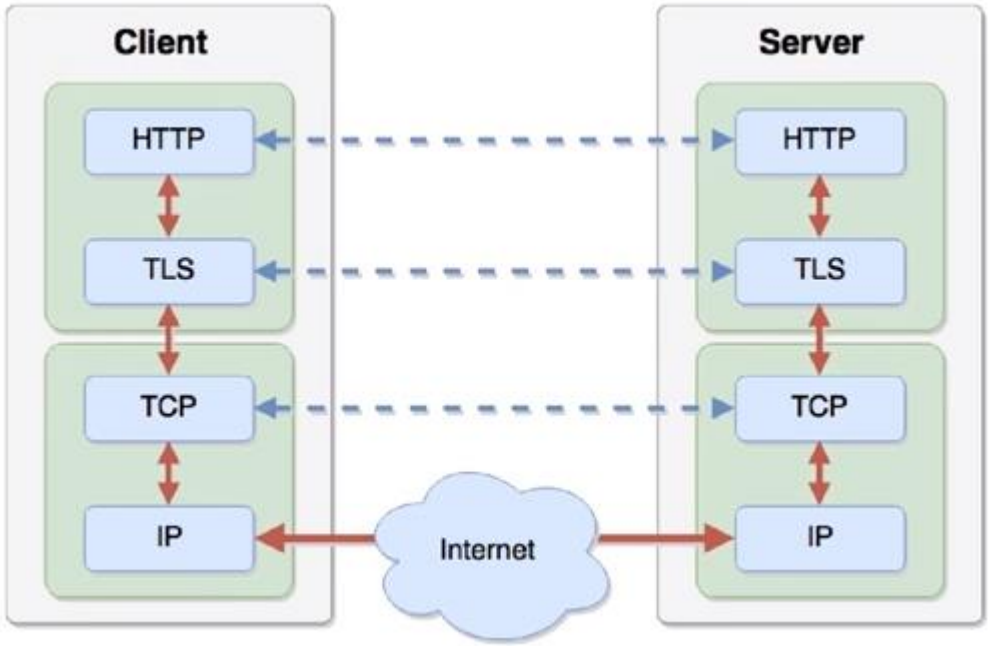
Анализ введенного URL и извлечение имени хоста

Преобразование системы DNS в доменное имя и адрес web-сервера

Установка TCP –соединение

Опциональная установка TSL соединения

Загрузка связанных ресурсов



Структура URI

Ресурс – это сетевой объект данных или сервис, который может быть идентифицирован URI. Ресурс можно запросить, получив ответ в виде файла (веб-страницы, медиафайла, документа). Ресурсы могут быть доступны в нескольких представлениях (на разных языках, в разных форматах, иметь различный размер и разрешающую способность).

URI определяется как строка, которая идентифицирует ресурс через имя, расположение или любую другую характеристику.

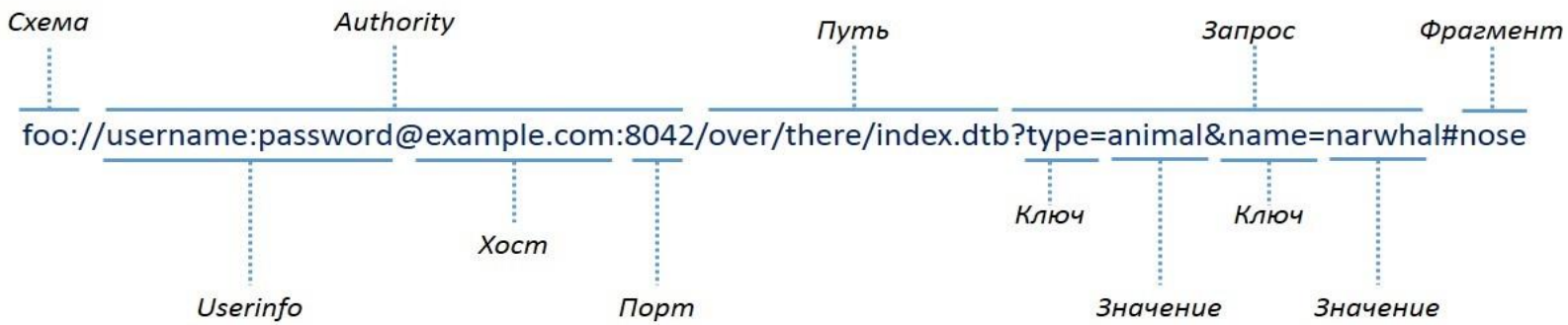
URI может содержать как местоположение ресурса (URL), так и его имя (URN).

В частном случае URI может означать имя файла или папки в файловой системе сервера, например:

http://www.lib.ru/WEBMASTER/rfc2068/section-5.html

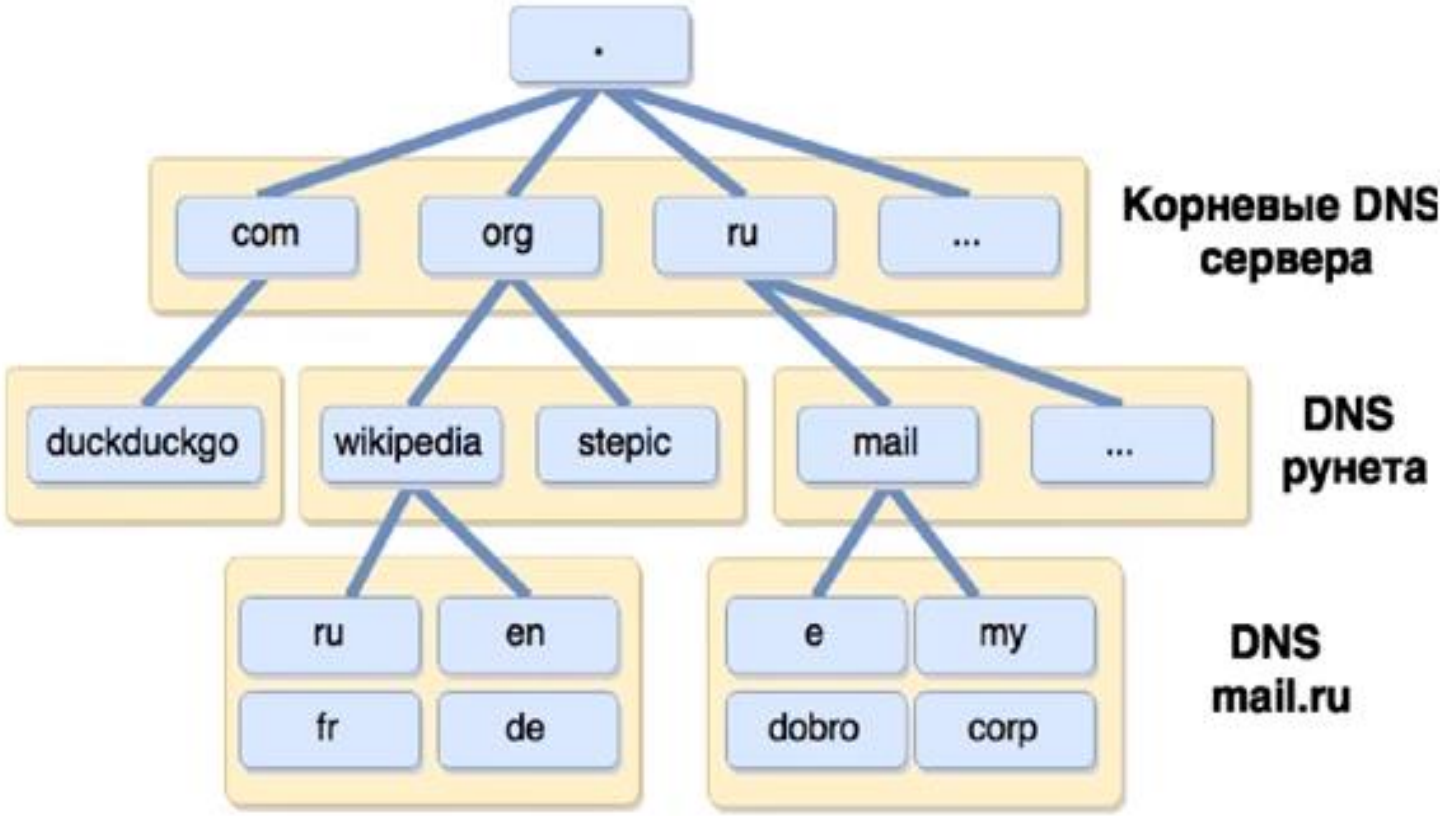
При этом необходимо понимать, что идентификация ресурса может происходить любым понятным серверу способом. Установление соответствия между URI и содержимым ресурса относится к компетенции веб-сервера, но не протокола HTTP.

Структура URI

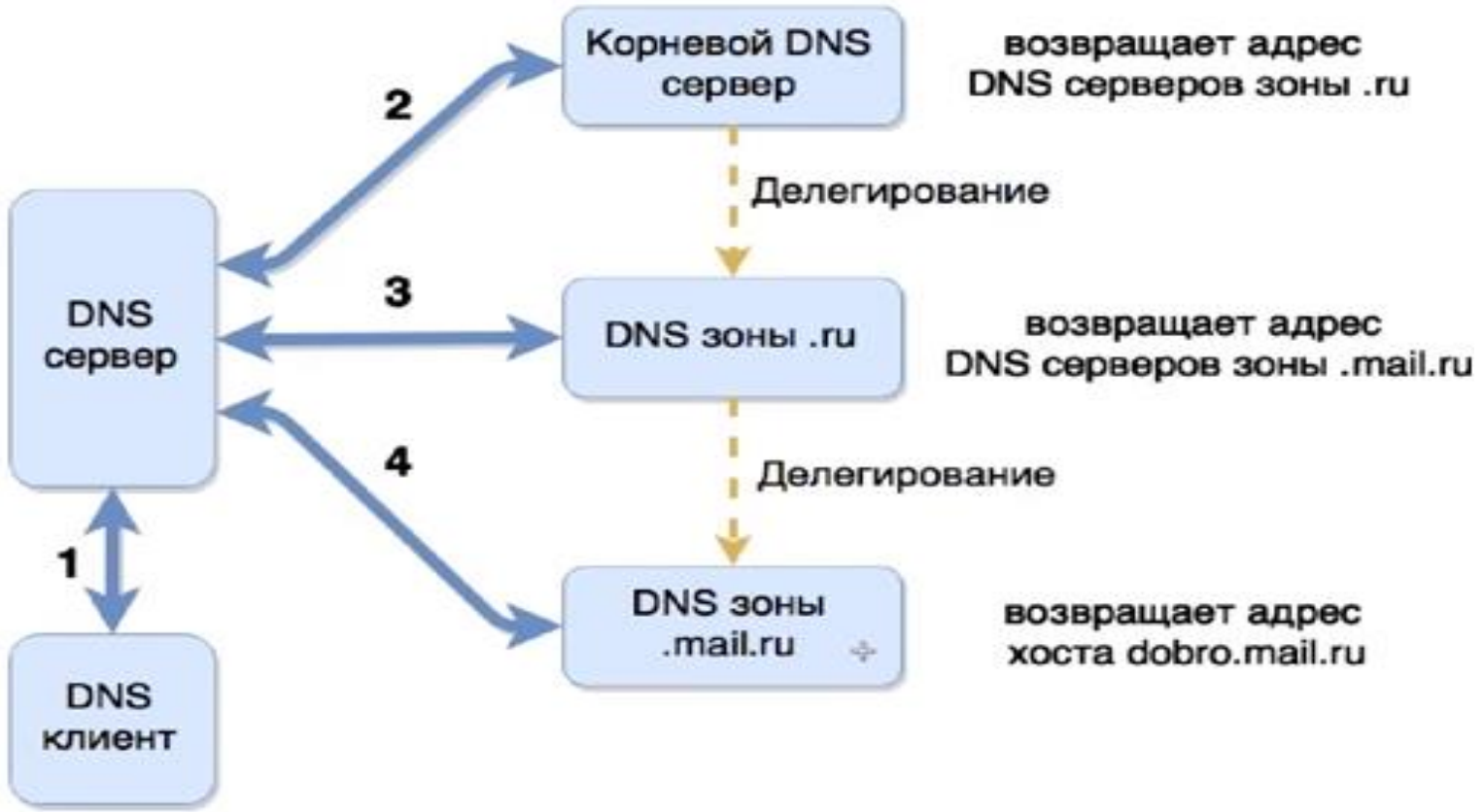


HEADLINE

DNS (англ. **Domain Name System** — система доменных имён) — компьютерная распределённая система для получения информации о доменах



HEADLINE


Корневой DNS сервер
возвращает адрес DNS серверов зоны .ru
2
Делегирование
DNS сервер
3
DNS зоны .ru
возвращает адрес DNS серверов зоны .mail.ru
4
Делегирование
1
DNS зоны .mail.ru
возвращает адрес хоста dobro.mail.ru
DNS клиент

HTTP-запрос

Запрос содержит:

- метод запроса
- URI
- версию протокола

} Строка запроса

- MIME-подобное сообщение с модификаторами запроса
- клиентскую информацию
- тело запроса (необязательно)

Пример

```
GET / HTTP/1.1
Accept: text/html,application/xhtml+xml,application/xml;q=0.9, */*;q=0.8
Accept-Encoding: gzip, deflate
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.5,en;q=0.3
Cache-Control: max-age=0
Connection: keep-alive
Cookie: yandexuid=3233555161427631248
Host: ya.ru
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64; rv:34.0) Gecko/20100101
  Firefox/34.0
```

Заголовок запроса

- Host – доменное имя и порт хоста запрашиваемого ресурса;
- Referer – URI ресурса, с которого клиент сделал текущий запрос;
- User-Agent – список названий и версий клиента и его компонентов с комментариями;
- Accept - список допустимых форматов ресурса по стандарту MIME;
- Accept-Charset – кодировка (utf-8, ascii);
- Accept-Encoding – кодирование содержимого (gzip и т. п.);
- Accept-Language – язык;
- Authorization – логин и пароль для входа;
- From – e-mail ответственного лица со стороны клиента;
- Proxy-Authorization – информация для авторизации на прокси-сервере.
- Range - диапазон для запроса фрагментов ресурса

Заголовок ответа

- Server - список названий и версий веб-сервера и его компонентов;
- WWW-Authenticate - параметры аутентификации для выполнения метода к указанному ресурсу;
- Age - количество секунд с момента модификации ресурса;

Заголовок сущности

- Content-Encoding, Content-Language, Content-Range, Content-Type – аналогично группе полей Accept;
- Content-MD5 – хеш для проверки целостности;
- Expires - дата предполагаемого истечения срока актуальности сущности;
- Last-Modified – дата последней модификации сущности;

HTTP- ответ

Запрос содержит:

- Версия протокола
- Код состояния
- Поясняющая фраза
- Поля заголовка
- тело запроса (необязательно)

Строка состояния

```
Пример
HTTP/1.1 200 OK
Server: nginx/0.6.31
Content-Language: ru
Content-Type: text/html; charset=utf-8
Content-Length: 1234
Connection: close

... САМА HTML-СТРАНИЦА ...
```

Проблемы безопасности

Проблемы безопасности, связанные с использованием HTTP, можно разделить на группы:

- проблемы, связанные с передачей данных;
- уязвимости клиентских приложений;
- уязвимости серверного программного обеспечения:
  - уязвимости серверов (Apache, nginx);
  - уязвимости в серверных сценариях (PHP, ASP).

Проблемы, связанные с передачей данных

Запрос проходит через множество сетей, любая из которых может быть использована для прослушивания или вмешательства в соединение.

- HTTP не использует шифрование, так как:
  - шифрование требует больше вычислительных мощностей;
  - при шифровании передаётся больше данных;
  - шифрованные страницы не кэшируются.

Пример: Когда пользователь просматривает web, злоумышленник может прослушивать незашифрованное Интернет-соединение пользователя, такое как HTTP (наиболее актуально для беспроводных сетей).

Для ослабления таких угроз применяют HTTPS, но наличие хотя бы одного небезопасного запроса сводит эффект к нулю.

Активные адресные атаки

- Активную атаку возможно реализовать с помощью подмены DNS-сервера, либо, в случае беспроводной сети, посредством фальсификации сетевых кадров или путем предоставления вредоносных точек доступа.
- Если пользователь находится позади беспроводного локального маршрутизатора, злоумышленник может попытаться реконфигурировать маршрутизатор, используя пароль по умолчанию и другие уязвимости.

Неадресные угрозы

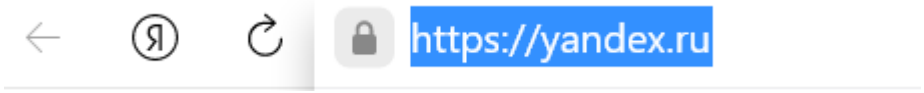
- Фишинг: злоумышленник просит аутентификационные параметры пользователя с помощью сайта, маскирующегося под доверенный ресурс. Атаки фишинга могут быть очень эффективными, так как для пользователей бывает трудно отличить фальшивый сайт от настоящего

Ошибки веб приложений

- Безопасность даже HTTPS-сайта может быть полностью скомпрометирована злоумышленником, использующим простую ошибку, такую как разрешение загрузки каскадного стилевого списка (CSS) или флэш-объекта (SWF), либо используя уязвимости PHP-Include, SQL-injection, Cross-Site Scripting (XSS).

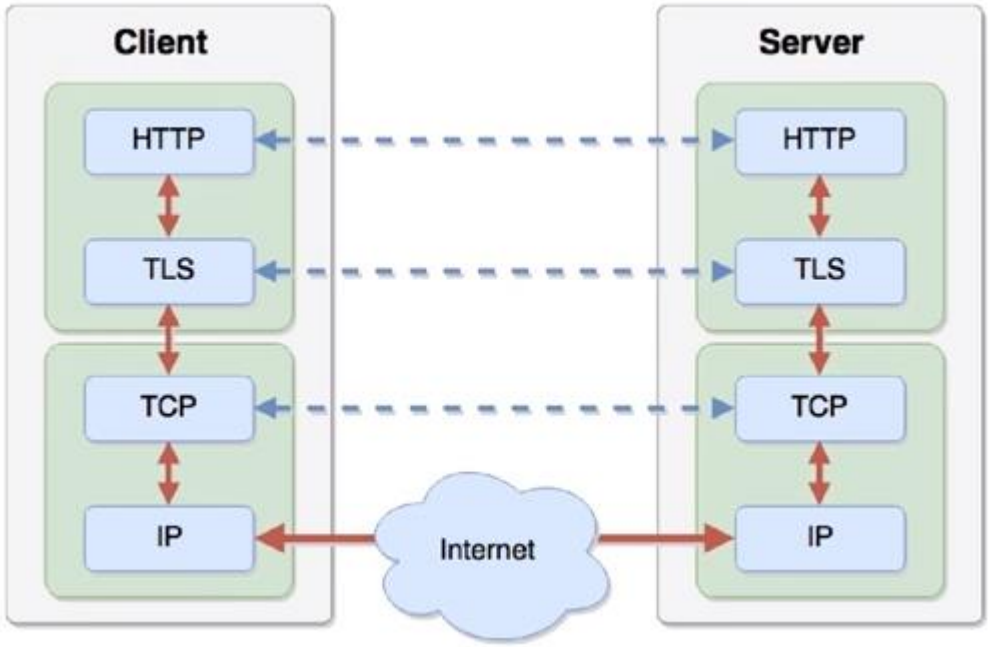
Протокол HTTPS

- HTTPS = HTTP + TLS/SSL
- HTTPS обеспечивает защиту от атак, основанных на прослушивании, при условии, что сертификат сервера проверен и используется шифрование.
- В отличие от HTTP, для HTTPS по умолчанию используется TCP-порт 443.
- Для организации сеанса HTTPS используются серверные сертификаты и, редко, клиентские сертификаты, которые позволяют реализовать взаимную аутентификацию.



Протокол TLS

- TLS (Transport Layer Security) - протокол, наиболее часто применяемый для обеспечения безопасного HTTP соединения.
- В модели протокол TLS размещается между прикладным протоколом и стеком TCP/IP.
- TLS использует:
  - асимметричную криптографию для аутентификации;
  - симметричное шифрование для конфиденциальности;
  - коды аутентичности сообщений (имитовставку) для контроля целостности данных.



Протокол HTTPS

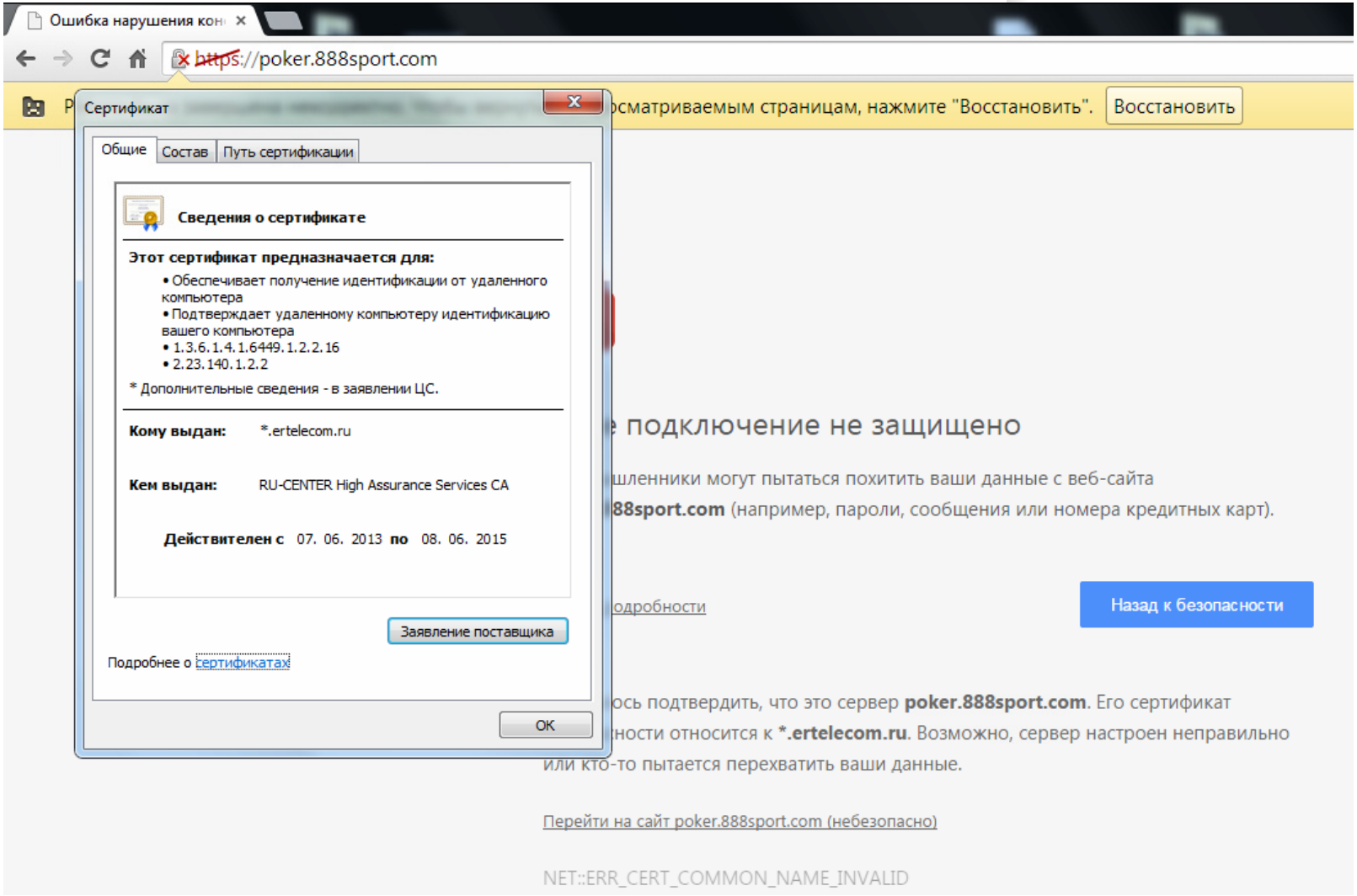
Идентификация сервера

1. После установления соединение, сервер посылает клиенту свой сертификат, чтобы клиент идентифицировал его. Это позволяет предотвратить атаку посредника. В сертификате указывается URI сервера.
2. Если имя сервера не совпадает с указанным в сертификате, то пользовательские программы, например браузеры, сообщают об этом пользователю. В основном, браузеры предоставляют пользователю выбор: продолжить незащищённое соединение или прервать его.

Идентификация клиента

Обычно сервер не располагает информацией о клиенте, достаточной для его идентификации. Однако для обеспечения повышенной защищённости соединения используется так называемая two-way authentication. При этом сервер после подтверждения его сертификата клиентом также запрашивает сертификат. Таким образом, схема подтверждения клиента аналогична идентификации сервера.

Проблемы HTTPS. Подмена сертификата



Проблемы HTTPS

- Если на компьютере пользователя присутствует троянская программа, имеющая доступ к браузеру, то HTTPS оказывается бесполезным, так как данные могут копироваться вовне до того, как попадут в зашифрованный канал.
- Если атакующая сторона получила соответствующий сеансовый ключ, то она может раскрыть HTTPS-трафик. Секретный серверный ключ или SSL-сертификат для этого не требуются.
- TLS (следовательно, и HTTPS) использует для реализации защищённого канала самые разные наборы криптографических алгоритмов (шифров, подписей, кодов аутентификации, дайджестов и так далее). Если какая-либо часть этого набора выбрана неверно, то соединение будет уязвимым, вне зависимости от того, какие ключи и SSL-сертификаты использовались (пример – уязвимость FREAK).
- Перехват HTTPS-соединения может быть автоматизирован – существуют специальные узлы-прокси, которые выполняют такой перехват на лету, в том числе, генерируя нужные сертификаты