ОБРПО. Лекция 5 HTTPS

ТЮРИН КАЙ АНДРЕЕВИЧ

HTTPS

HTTPS: HyperText Transfer Protocol Secure — расширение протокола HTTP для поддержки шифрования в целях повышения безопасности.

HTTPS работает поверх SSL/TLS

(TLS1.0 почти то же, что SSL3.1)

Стандартный порт для HTTP – 80.

Для HTTPS – 443.

В этой лекции нет пересказа спеки

Тем более, что для подробного изучения внутренностей протокола TLS (на котором основан HTTPS) есть великолепный сайт «The Illustrated TLS Connection»

https://tls.ulfheim.net/

❧ The Illustrated TLS Connection ❧

Every byte of a TLS connection explained and reproduced.

In this demonstration a client connects to a server, negotiates a TLS 1.2 session, sends "ping", receives "pong", and then terminates the session. Click below to begin exploring.

⇉ Client Hello

⇇ Server Hello

⇇ Server Certificate

⇉ Client Hello ×

The session begins with the client saying "Hello". The client provides the following:

- protocol version
- client random data (used later in the handshake)
- an optional session id to resume
- a list of cipher suites
- a list of compression methods
- a list of extensions

Annotations

```
16 03 01 00 a5
```

Record Header

TLS sessions are broken into the sending and receiving of "records", which are blocks of data with a type, a protocol version, and a length.

- `16` - type is 0x16 (handshake record)
- `03 01` - protocol version is 3.1 (also known as TLS 1.0)
- `00 a5` - 0xA5 (165) bytes of handshake message follows

Вспомним криптографию

$$E_k(p) = c \qquad D_k(c) = p$$

$$E_{pk}(p) = c \qquad D_{sk}(c) = p$$

$$Sign_{sk}(m) = s \qquad Verify_{pk}(m, s) = 0/1$$

Этапы соединения (1)

1) Начало соединения

2) Клиент предоставляет список алгоритмов шифрования. Сервер выбирает наиболее надежный алгоритм и предлагает общаться по нему

3) Сервер отправляет клиенту свой цифровой сертификат, подписанный CA, и открытый ключ сервера.

Этапы соединения (2)

4) Клиент проверяет сертификат (подключившись к CA или воспользовавшись имеющимся сертификатом

5) Генерируется сеансовый ключ. Это делается, например, так:
— Клиент генерирует случайную последовательность
— Клиент шифрует ее открытым ключом сервера и посылает результат на сервер
— Сервер расшифровывает полученную последовательность при помощи закрытого ключа

6) Соединение установлено

(А)Симметричное шифрование

Вообще-то асимметричное шифрование достаточно медленное

Поэтому его часто используют для генерации ключа симметричного шифрования

Такой ключ называется сессионным

Что пытается мы хотим защитить?

1. Данные, передаваемые по сети
    SSL/TLS
2. Код/данные в браузере (cookie и т.д.)
    https://
    Имя хоста в сертификате
    Замок в браузере
3. UI, который видит пользователь

Certificate Authority (CA)

Подписывает ключи различных доменов

$$Sign_{CA's\ secret\ key}(domain\ name: domain\ puplic\ key) = s$$

Так, пользователь может проверить подлинность ключа

$$Verify_{CA's\ public\ key}(domain\ name: domain\ puplic\ key, s) = 0/1$$

Таким образом, CA не обязательно должен всегда быть онлайн

(пользователь тоже может быть аутентифицирован)

Откуда берутся CA сертификаты?

Сертификаты установлены по умолчанию в браузеры, операционные системы и т.д.

CA очень много.

text 5.50 KB

```
Hello

I'm writing this to all the world, so you'll know more about us..

At first I want to give some points, so you'll be sure I'm the hacker:

I hacked Comodo from InstantSSL.it, their CEO's e-mail address mfpenco@mfpenco.com
Their Comodo username/password was: user: gtadmin password: globaltrust
Their DB name was: globaltrust and instantsslcms

Enough said, huh? Yes, enough said, someone who should know already knows...
```

CA сертификаты вечны?

Они имеют ограничение срока действия.

Кроме того, необходимы механизмы отзыва сертификатов.

Такими механизмами являются:

CRL

OCSP

CRL

Certificate Revocation List

Список отозванных сертификатов (COC).

Содержит только неистёкшие сертификаты.

Клиентам приходится качать эти списки… или не качать.

OCSP

Online Certificate Status Protocol

Предназначен для проверки валидности сертификата запросом к OCSP серверу

Проблемы:

1. Добавление задержки для соединений

2. Если OCSP сервер лежит, значит… будем считать сертификат валидным!

Что на самом деле используется?

Хардкод плохих сертификатов!

Прямо в браузере!

Cookie secure

Cookie, доступные из HTTP и HTTPS:

```
Set-Cookie: id=a3fWa;
```

Cookie, доступные только из HTTPS:

```
Set-Cookie: id=a3fWa; Secure
```

Небезопасная зависимость

```
<script src="http://example.com/example.js"
    ></script>
```

Внедрение в зависимости от схемы

```
<script src="//example.com/example.js"
    ></script>
```

Subresource Integrity (SRI)

```
<script src="https://example.com/example.js"
        integrity="sha384-
oqVuAfXRKap7fdgcCY5uykM6+R9GqQ8K/uxy9rx7HNQlG
Yl1kPzQho1wx4JwY8wC"></script>
```

HSTS

1. Ошибки сертификата – фатальны для соединения
2. Редирект с HTTP на HTTPS
3. Запрет не-HTTPS вложений

Проблемы:

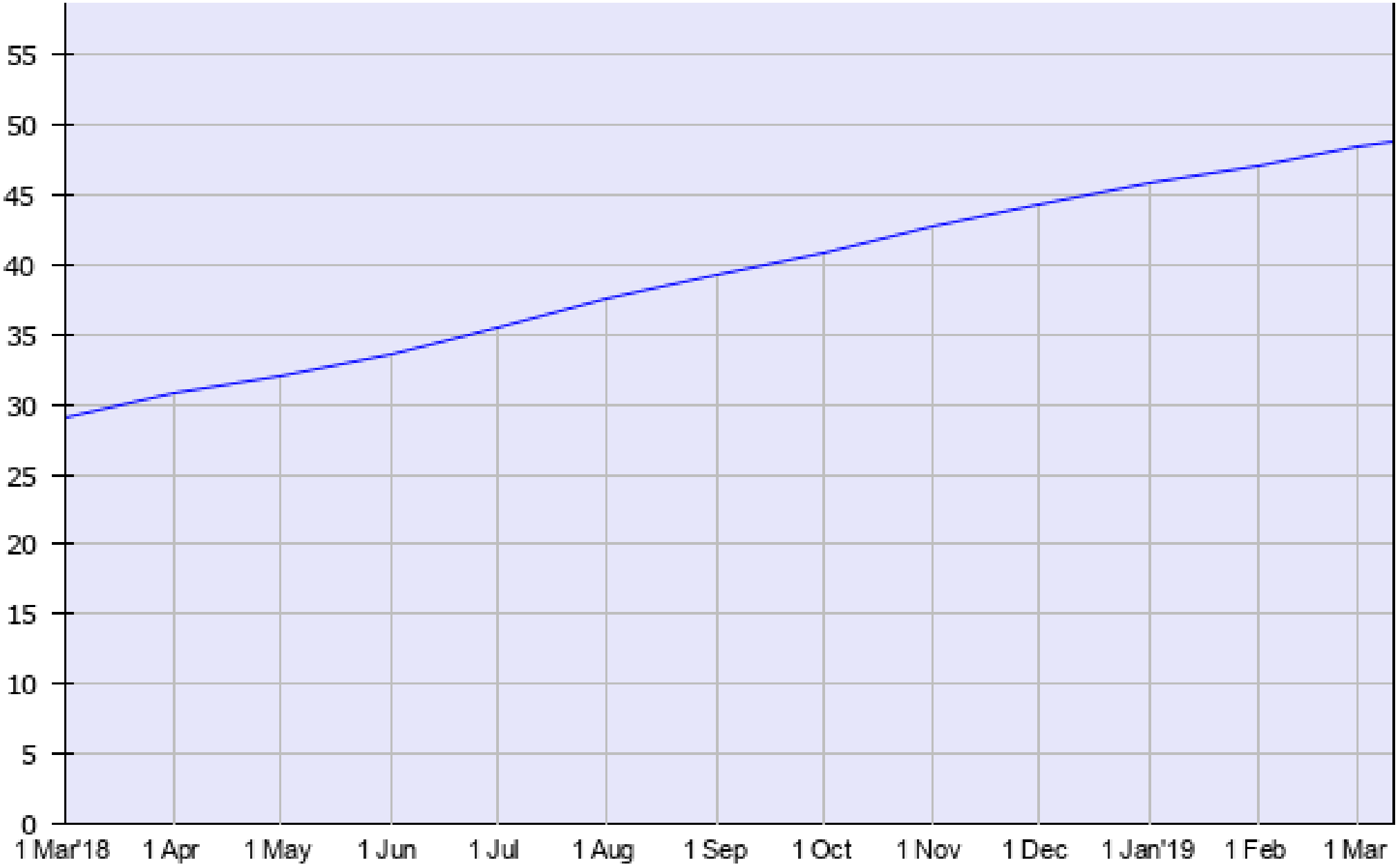
1. Первое соединение
2. Возможность DoS атак

DNS в контексте HTTPS

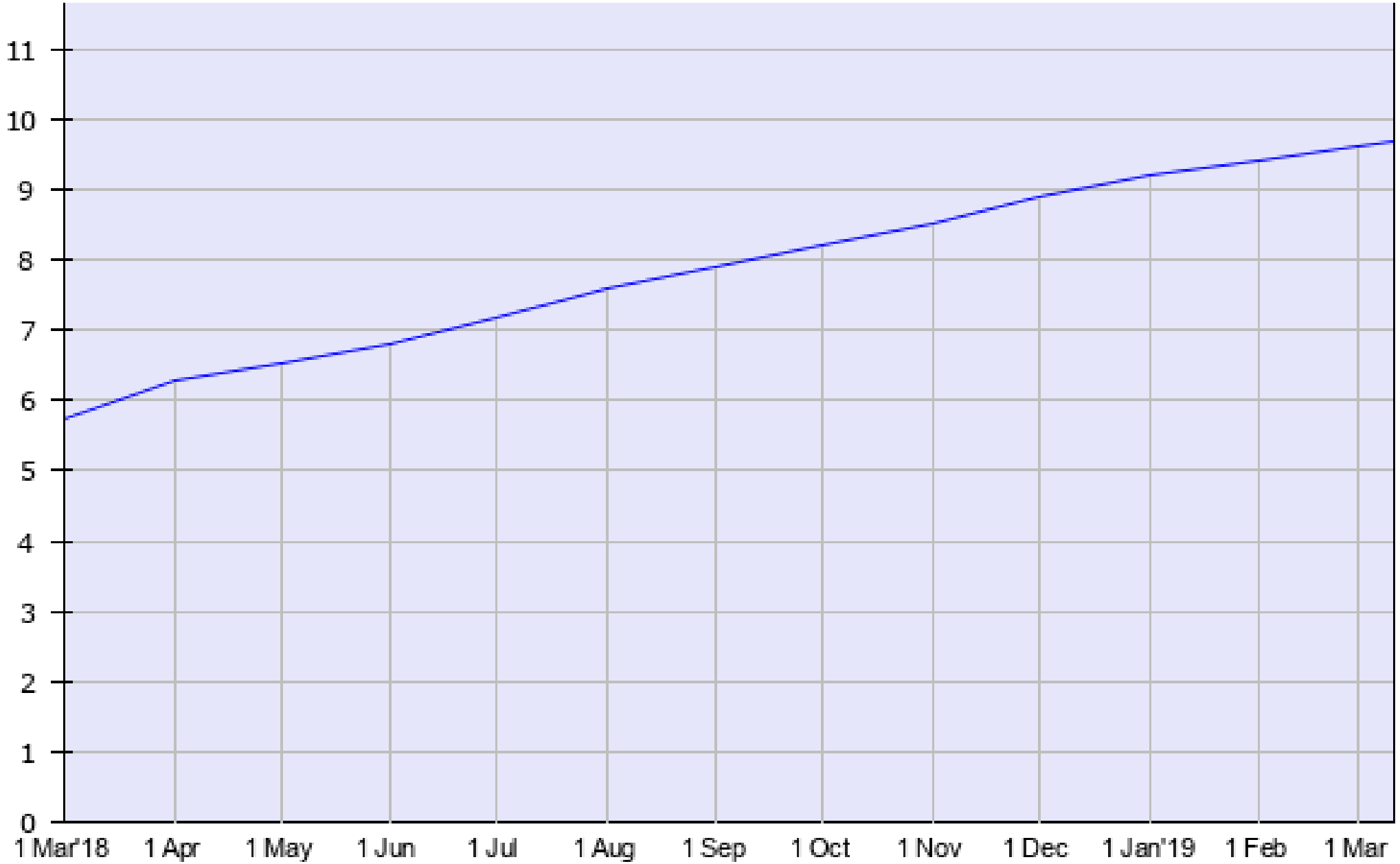
DNS всё ещё нужен, чтобы находить IP адрес по имени домена

Тем не менее, сервер должен иметь валидный сертификат, подписанный CA.

Таким образом, HTTPS обеспечивает безопасность в независимости от DNS.



Usage of Default protocol https for websites, 11 Mar 2019, W3Techs.com



Usage of HTTP Strict Transport Security for websites, 11 Mar 2019, W3Techs.com