

# Основы построения защищенных компьютерных сетей

Лекция 10  
Криптографические протоколы. TLS



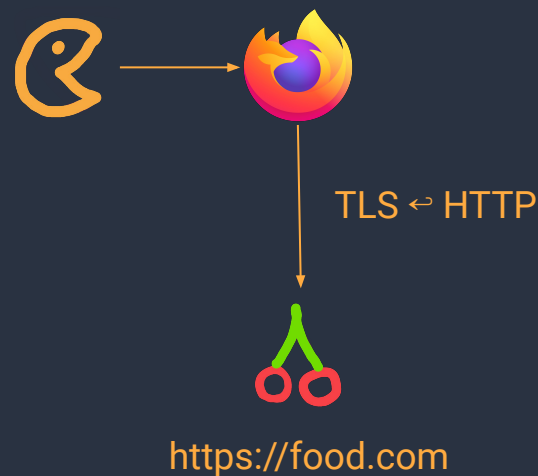
Семён Новосёлов

2023

**БФУ**  
ИМЕНИ И. КАНТА

# Протокол TLS (Transport Layer Security)

- Широко используемый протокол для защищенной передачи данных между узлами
- Подключение к большинству сайтов в настоящее время осуществляется по TLS



# Версии TLS/SSL

## SSL (Secure Sockets Layer)

- SSLv1: взломан сразу же после появления
- SSLv2: устарел (RFC6176), имеет слабости
  - использование MD5
  - атаки MiTM с навязыванием слабых наборов шифров
- SSLv3: взломан (POODLE + атаки на RC4)

## TLS (Transport Layer Security) — стандартизованная версия SSL

- TLS 1.0 / TLS 1.1: устарели (RFC8996)
- TLS 1.2: безопасность зависит от набора шифров и мер защиты у клиента
- TLS 1.3: действующая версия

# Криптоалгоритмы в TLS

- Цифровые подписи для проверки подлинности сервера
- Протокол Диффи-Хеллмана для выработки общего ключа
- Симметричное шифрование с выработанным общим ключом для защиты данных
- Хэш-функции для проверки целостности сообщений

# Схема работы

1. Установка соединения (**handshake**)
  - 1.1. Согласование наборов криптоалгоритмов
  - 1.2. Проверка подлинности сторон
  - 1.3. Выработка сессионного ключа
2. Передача данных
  - 2.1. С шифрованием данных по сессионному ключу

# Установка соединения (handshake)

1. Клиент подключается к серверу и посылает список поддерживаемых криптоалгоритмов:
  - 1.1. цифровые подписи
  - 1.2. хэш-функции
  - 1.3. симметричные шифры
  - 1.4. схемы обмена ключами
2. Сервер выбирает набор криптоалгоритмов и уведомляет клиента о выбранном наборе

# Пример

No.	Time	Source	Destination	Protocol	Length	Info
303	18.844421	192.168.1.34	dyna.wikimedia.org	TCP	66	2052 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS
305	18.920772	dyna.wikimedia.org	192.168.1.34	TCP	66	443 → 2052 [SYN, ACK] Seq=0 Ack=1 Win=4234
306	18.920826	192.168.1.34	dyna.wikimedia.org	TCP	54	2052 → 443 [ACK] Seq=1 Ack=1 Win=132096 Len
307	18.921119	192.168.1.34	dyna.wikimedia.org	TLSv1.3	571	Client Hello
308	18.996146	dyna.wikimedia.org	192.168.1.34	TCP	54	443 → 2052 [ACK] Seq=1 Ack=518 Win=41984 Len
309	18.997409	dyna.wikimedia.org	192.168.1.34	TLSv1.3	1414	Server Hello, Change Cipher Spec, Applicat
310	18.997593	dyna.wikimedia.org	192.168.1.34	TCP	1414	443 → 2052 [ACK] Seq=1361 Ack=518 Win=4249
311	18.997593	dyna.wikimedia.org	192.168.1.34	TLSv1.3	1182	Application Data, Application Data, Applic

Cipher Suites Length: 32

▼ Cipher Suites (16 suites)

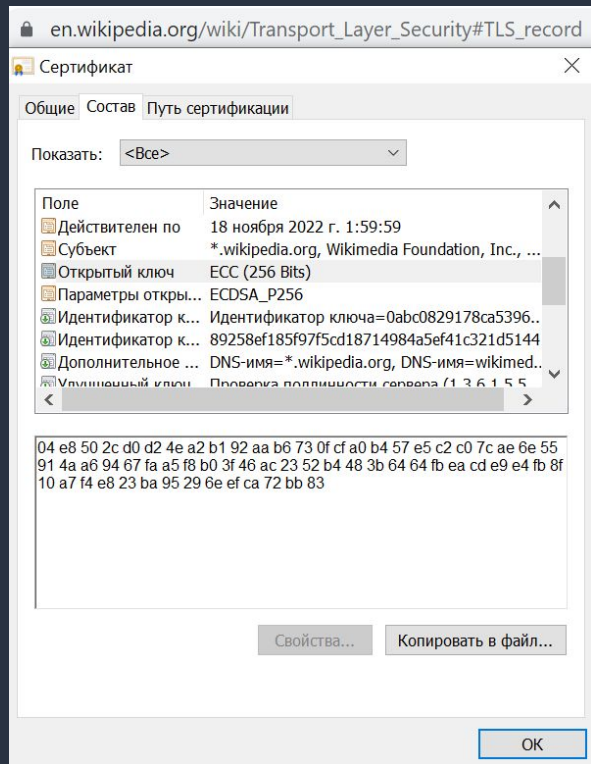
- Cipher Suite: Reserved (GREASE) (0x6a6a)
- Cipher Suite: TLS\_AES\_128\_GCM\_SHA256 (0x1301)
- Cipher Suite: TLS\_AES\_256\_GCM\_SHA384 (0x1302)
- Cipher Suite: TLS\_CHACHA20\_POLY1305\_SHA256 (0x1303)
- Cipher Suite: TLS\_ECDHE\_ECDSA\_WITH\_AES\_128\_GCM\_SHA256 (0xc02b)
- Cipher Suite: TLS\_ECDHE\_RSA\_WITH\_AES\_128\_GCM\_SHA256 (0xc02f)
- Cipher Suite: TLS\_ECDHE\_ECDSA\_WITH\_AES\_256\_GCM\_SHA384 (0xc02c)

# Handshake. Сертификаты

3. Сервер также посылает свой сертификат:  
имя сервера + открытый ключ сервера + имя центра сертификации (CA)  
+ подпись (CA)
4. Клиент проверяет полученный сертификат, используя встроенную базу ключей CA



# Пример. Сертификат Википедии



# Handshake. Выработка сессионного ключа

4. Если сертификат сервера действителен  $\Rightarrow$  выполняется один из вариантов:
  - 4.1. клиент и сервер вырабатывают сессионный ключ по протоколу Диффи-Хеллмана
  - 4.2. клиент выбирает случайное число  $\Rightarrow$  шифрует его по открытому ключу сервера  $\Rightarrow$  посылает серверу  $\Rightarrow$  сессионный ключ вырабатывается на основе случайного числа

# Передача данных

- После выработки общего сессионного ключа данные передаются с использованием симметричного шифрования.

```
14... 14.048202 dyna.wikimedia.org      10.252.65.211      TLSv1.3      357 Application Data
<
> Transmission Control Protocol, Src Port: 443, Dst Port: 1302, Seq: 597, Ack: 810, Len: 31
v Transport Layer Security
  v TLSv1.3 Record Layer: Application Data Protocol: http-over-tls
    Opaque Type: Application Data (23)
    Version: TLS 1.2 (0x0303)
    Length: 26
    Encrypted Application Data: 549ad251a54ea082329097dd88d42ccb9c23df5cc4f47a52cbfe
    [Application Data Protocol: http-over-tls]
<
0030  00 53 7d 0b 00 00 17 03  03 00 1a 54 9a d2 51 a5  .S}... ..T..Q.
0040  4e a0 82 32 90 97 dd 88  d4 2c cb 9c 23 df 5c c4  N..2... .,.,#\.
```

# Уязвимости и атаки в SSL/TLS

- POODLE (CVE-2014-3566)
- Heartbleed (CVE-2014-0160)
- Logjam (CVE-2015-4000)
- DROWN (CVE-2016-0800)
- Cloudbleed (2017)
- ...

# Уязвимость Heartbleed (утечка памяти)

## Где?

OpenSSL: широко распространённая библиотека

## Что утекает?

кусочки по 64 Кб из зашифрованных соединений



## Кого затронуло?

- Yandex
- Google
- банковские сайты
- (всех)

# TLS. Расширение Heartbeat

**Появление:** февраль 2012 года

**Функция:** поддержание соединения в открытом состоянии (keep-alive)

**Принцип работы:**

- периодически посылаются пакеты с текстовой строкой
- сервер посылает в ответ пакет с той же самой строкой



# Схема уязвимости



# Уязвимый код

считывание размера  
полезной нагрузки из  
пришедшего пакета

```
unsigned int payload;
unsigned int padding = 16; /* Use minimum padding */
/* Read type and payload length first */
hbtype = *p++;
n2s(p, payload);
p1 = p;
//...
/* Allocate memory for the response, size is 1 byte
 * message type, plus 2 bytes payload length, plus
 * payload, plus padding */
buffer = OPENSSL_malloc(1 + 2 + payload + padding);
bp = buffer;
```

```
/* Enter response type, length and copy payload */
*bp++ = TLS1_HB_RESPONSE;
s2n(payload, bp);
memcpy(bp, p1, payload);
bp += payload;
/* Random padding */
RAND_pseudo_bytes(bp, padding);
r = dtls1_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, 3
+ payload + padding);
```



# Уязвимый код

выделение памяти под ответный пакет.  
**Ошибка:** нет проверки входных данных (payload).

```
unsigned int payload;
unsigned int padding = 16; /* Use minimum padding */
/* Read type and payload length first */
hbtype = *p++;
n2s(p, payload);
p1 = p;
//...
/* Allocate memory for the response, size is 1 byte
 * message type, plus 2 bytes payload length, plus
 * payload, plus padding */
buffer = OPENSSL_malloc(1 + 2 + payload + padding);
bp = buffer;
```

```
/* Enter response type, length and copy payload */
*bp++ = TLS1_HB_RESPONSE;
s2n(payload, bp);
memcpy(bp, p1, payload);
bp += payload;
/* Random padding */
RAND_pseudo_bytes(bp, padding);
r = dtls1_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, 3
+ payload + padding);
```

# Уязвимый код

копирование присланной нагрузки Heartbeat в пакет-ответ  
**Ошибка:** если прислать меньше данных, чем указано в payload, то копируются данные в памяти, лежащие после buffer

```
unsigned int payload;
unsigned int padding = 16; /* Use minimum padding */
/* Read type and payload length first */
hbtype = *p++;
n2s(p, payload);
p1 = p;
//...
/* Allocate memory for the response, size is 1 byte
 * message type, plus 2 bytes payload length, plus
 * payload, plus padding */
buffer = OPENSSL_malloc(1 + 2 + payload + padding);
bp = buffer;
```

```
/* Enter response type, length and copy payload */
*bp++ = TLS1_HB_RESPONSE;
s2n(payload, bp);
memcpy(bp, p1, payload);
bp += payload;
/* Random padding */
RAND_pseudo_bytes(bp, padding);
r = dtls1_write_bytes(s, TLS1_RT_HEARTBEAT, buffer, 3
+ payload + padding);
```

# Уязвимый код

Формирование выходного пакета для последующей отправки данных

```
unsigned int payload;
unsigned int padding = 16; /* Use minimum padding */
/* Read type and payload length first */
hbtype = *p++;
n2s(p, payload);
p1 = p;
//...
/* Allocate memory for the response, size is 1 byte
 * message type, plus 2 bytes payload length, plus
 * payload, plus padding */
buffer = OPENSSL_malloc(1 + 2 + payload + padding);
bp = buffer;
```

```
/* Enter response type, length and copy payload */
*bp++ = TLS1_HB_RESPONSE;
s2n(payload, bp);
memcpy(bp, p1, payload);
bp += payload;
/* Random padding */
RAND_pseudo_bytes(bp, padding);
r = dtls1_write_bytes(s, TLS1_RT_HEARTBEAT,
buffer, 3 + payload + padding);
```

# Что может утечь?

- секретные ключи TLS
- cookies
- логины
- пароли
- куски писем
- любые другие данные, которыми обменивается сервер и его клиенты



**Уязвимость двусторонняя:** сервер тоже может читать данные клиента

# Исправление

```
if (1 + 2 + payload + 16 > s->s3->rrec.length)
    return 0; /* silently discard per RFC 6520 sec. 4
*/
```

Добавление проверки с фактически полученным размером данных

# Как избежать подобных ошибок?

**Очевидно:** проверять корректность всех полученных извне данных.

Выполнять аудит кода с помощью инструментов **статического** и **динамического** анализа кода.

# Статический анализ

Проверка исходного кода без запуска программы.

**Недостатки:** Много ложных срабатываний и много времени тратится на анализ отчётов.

**Инструменты:**



# Динамический анализ кода

Анализ запущенной программы.

Для нахождения ошибок может использоваться следующая связка:





# Литература и ссылки

- Kenny Paterson. “TLS: A Real-World Secure Channel Protocol”  
<http://cyber.biu.ac.il/wp-content/uploads/2018/02/lecture1-3.pdf>

