Уральский федеральный университет

имени первого Президента России Б. Н. Ельцина

Кафедра «Интеллектуальные информационные технологии»

**ПРАКТИЧЕСКАЯ РАБОТА 2**

ОТЧЁТ

Дисциплина «Веб-программирование»

Проверил:

Свинцов Д. В.

Выполнил:

студент гр.

РИ-380022

Гасанов Г. Р.

Екатеринбург 2020

**Практическая работа № 2**

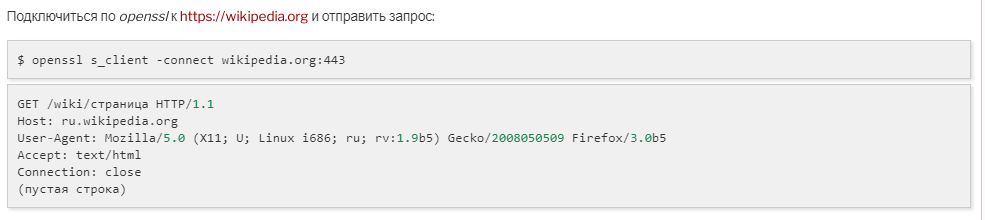
**Работа с протоколом HTTPS через opensll**

**1)Цель работы**

Получить практические навыки по работе с https протоколом посредством openssl. Научится создавать сертификаты.

**2) Задание 1.**

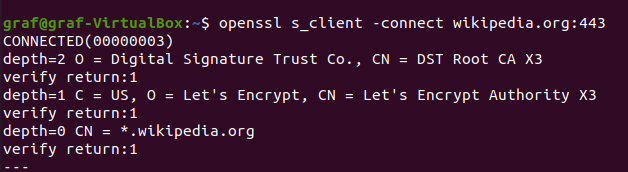
**Задание:**

 **Ход работы:**

Подключим по openssl к [https://wikipedia.org](https://wikipedia.org/). с помощью следующей команды:

$ openssl s\_client -connect wikipedia.org:443

В первых строчках ответа можно увидеть, что соединение было установлено, как показано на рисунке 1.

Рисунок 1 — Информация о серверном сертификате

Здесь отображаются содержимое цепочки сертификатов.(Рисунок 2).

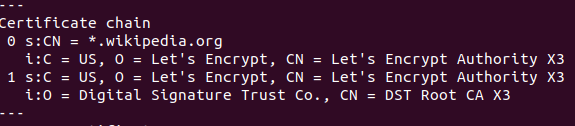


Рисунок 2 — Содержимое цепочки сертификатов

Затем выводится серверный сертификат, который начинается с ключевого слово BEGIN и заканчивается словом END. (Рисунок 3, Рисунок 4).



Рисунок 3 — Вывод серверного сертификата, начинающая с BEGIN

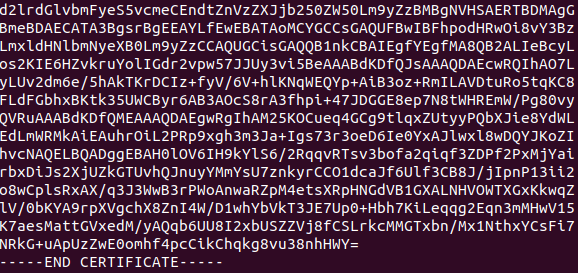


Рисунок 4 — Вывод серверного сертификата, заканчивающая на END

Дальше идёт назначение текста сертификата сервера subject и информация об издателе issuer.

 Информация о сертификате представлена на рисунке 5.

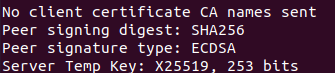


Рисунок 5 — Информация о сертификате

Вывод информации об используемом алгоритме шифрования и версии протокола (Рисунок 6).

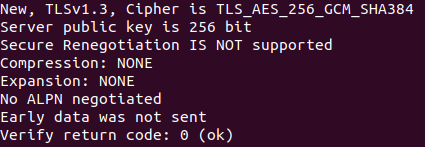
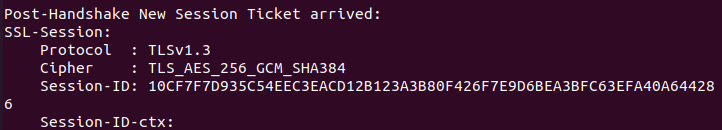
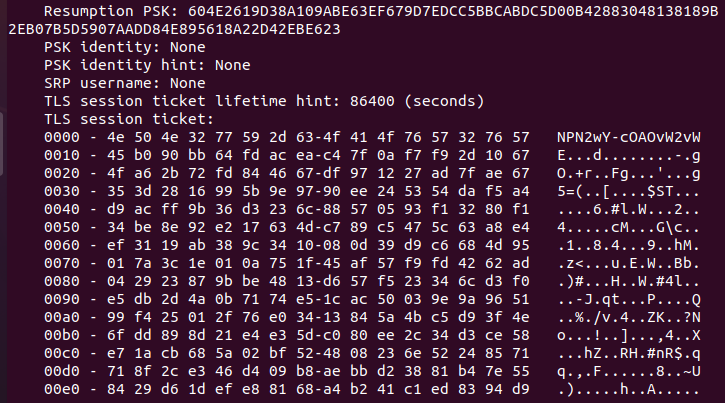


Рисунок 6 — Информация о TLS – соединении и об алгоритме шифрования AES\_256

Далее сервер выпустил для нас session\_id и TLS session ticket.(Рисунок 7, Рисунок 8).

Рисунок 7 — Session\_id

Рисунок 8 — TLS session ticket

Отправляем GET запрос (Рисунок 9).

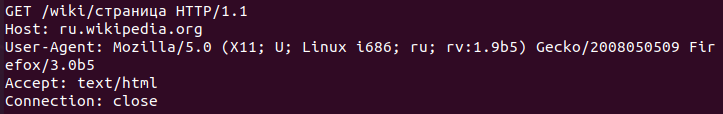


Рисунок 9 — GET – запрос

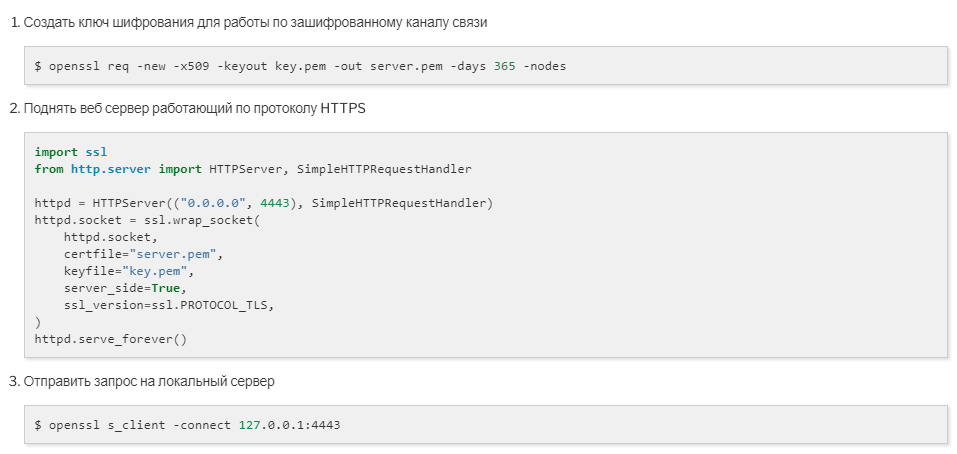
Ответ сервера (Рисунок 10).

Рисунок 10 — ответ сервера

В ответ получен код состояния 301. Также URL, который был запрошен, перенесен на адрес, хранящийся в поле Location.

**Задание 2.**

**Задание:**

**Ход работы:**

Создадим директорию и перейдём в ней через командую строку, как показано на рисунке 11.

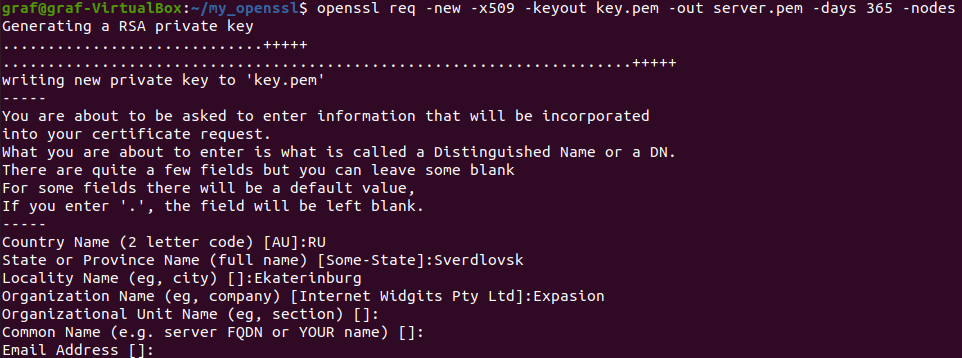


Рисунок 11 — Создание и переход в директорию

Выполним следующую команду:

$ openssl req -new -x509 -keyout key.pem -out server.pem -days 365 –nodes

В результате выполнения команды будут созданы два ключа. Один приватного использования, второй публичного, со сроком использования 365 дней. (Рисунок 12).

Рисунок 12 — Создание двух ключей

Поднимем веб-сервер, работающий по протоколу HTTPS. Для этого через командную строку создадим файл с расширением my\_server.py, который находится в директории my\_openssl.(Рисуно13).



Рисунок 13 — Создание файла в формате .py

Далее необходимо записать туда следующее, как показан на рисунке 14.

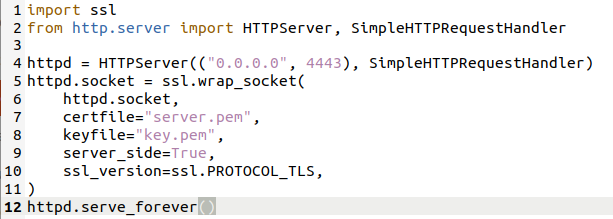


Рисунок 14 — Исходный код сервера

Запускаем файл, как продемонстрировано на рисунке 15.

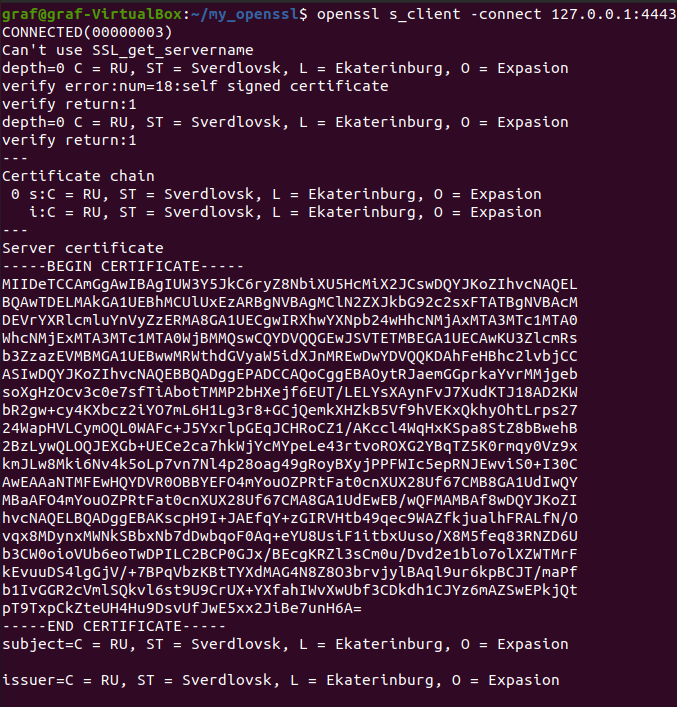


Рисунок 15 — Результат подключения к локальному серверу

**Задание 3.**

1. Введём следующую команду:

$ openssl s\_client –connect httpbin.org:443

Далее запросим данные GET запросом с ресурса ip

GET /ip HTTP/1.1

Host: httpbin.org

Accept: \*/\*

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 16 — Ответ сервера

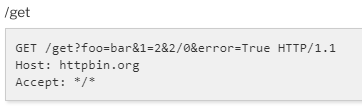
HTTP/1.1 200 OK – код состояния выполнения запроса, который означает, что выполнение прошло успешно

Content-Type: application/json – означает, что данные выводятся в формате json

Content-Length: 33 – длинна ответ 33 символа

“origin”: “178.176.122.149” – это тот ответ в формате json, длинною 34 символ

1. **Выполнить запрос методом GET**

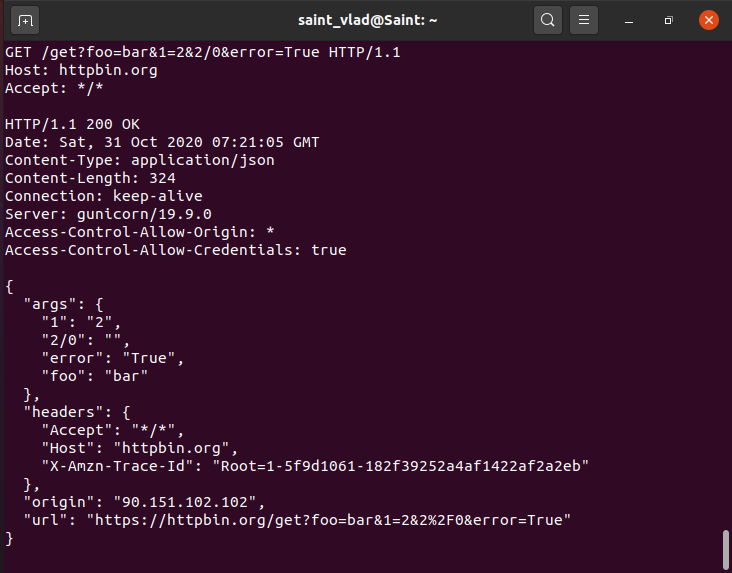
Отправляем запрос:

GET /get?foo=bar&1=2&2/0&error=True HTTP/1.1

Host: httpbin.org

Accept: \*/\*

Ответ сервера:

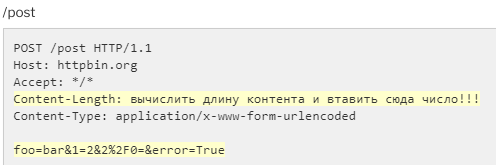
****

HTTP/1.1 200 OK – код состояния выполнения запроса, который означает, что выполнение прошло успешно

Content-Type: application/json – означает, что данные выводятся в формате json

Content-Length: 324 – длинна ответ 324 символа

1. **Выполнить запрос методом POST**



Ответ сервера:

**Изображение выглядит как текст

Автоматически созданное описание**

HTTP/1.1 200 OK – код состояния выполнения запроса, который означает, что выполнение прошло успешно

Content-Type: application/json – данные выводятся в формате json

Content-Length: 373 – длинна ответа 373 символа

1. **Отправить запрос на установку Cookie**



Ответ сервера

**Изображение выглядит как текст

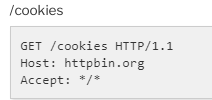
Автоматически созданное описание**

HTTP/1.1 302 FOUND – код состояния выполнения запроса, который означает, что ресурс был временно перемещён, адрес можно увидеть в Location

Set-Cookie: country=Ru; Path=/ – запрос был получен

Результатом служит перенаправление на другую страницу

1. **Посмотреть список установленных Cookie**



Изображение выглядит как текст

Автоматически созданное описание

Код состояния запроса 200. Запрос успешно выполнен