**Лабораторная работа №2**

**«Работа с протоколом HTTPS через opensll»**

Выполнил: Ширшов Игорь Алексеевич РИ-380022

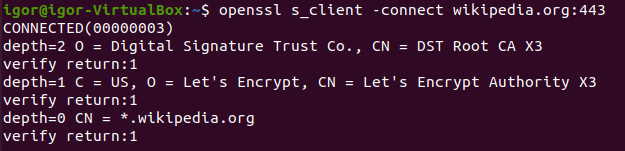
**Задание 1.**

Постановка задачи:

Изображение выглядит как текст

Автоматически созданное описание

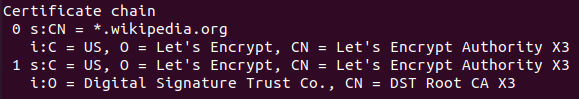
Описание выполнения работы:



Первая строка использует команду s\_client, предназначенную для установки соединения с удалённым SSL/TLS-сервером, приема результатов их выполнения и отображения информации об используемых параметрах SSL/TLS, после содержится единственный ключ, с помощью которого выполняется подключение к сайту wikipedia.org по порту 443.

Вторая строка подтверждает факт установки соединения.

В остальных строках содержится результат проверки цепочки сертификатов, использованной при установке соединения. depth – это глубина каждого сертификата в цепочке сертификатов, С – страна, О – организация, ОU – подразделение, СN – имя субъекта сертификации, и verify return – результат проверки.

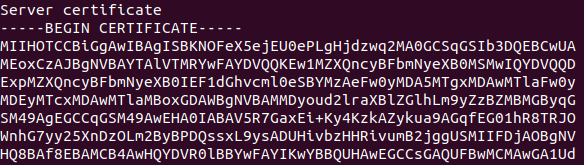


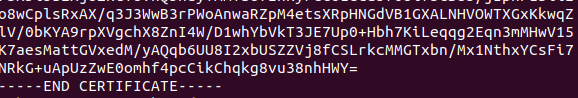
Здесь отображаются содержимое цепочки сертификатов, установленной на сервере (Certificate chain).

С номером 0 – это сертификат Википедии. У проверяемого сайта есть ещё один сертификат, он указан под номером 1.

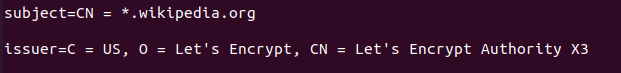
s – это назначение для каждого сертификата, которое содержит , С – страна, О – организация, СN – имя субъекта сертификации.

i – это информация об издателе.

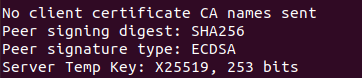




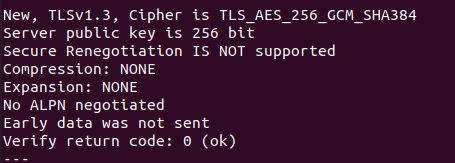
Выводимый текст сертификата сервера в формате PEM , который начинается с BEGIN и заканчивается END.



Назначение текста сертификата сервера subject и информация об издателе issuer.



У нас самоподписанный сертификат SSL, поэтому выводятся такие команды. Тут определен формат подписи: SHA256 ECDSA



В первой строке можно увидеть, что используется новый протокол TLSv1.3 и набор шифров Cipher is TLS\_AES\_256\_GCM\_SHA384. Это генерация сеансового ключа. Алгоритм AES256 – для шифрования трафика с длиной ключа 256 бит в режиме GCM, алгоритм SHA384 – для контроля целостности.

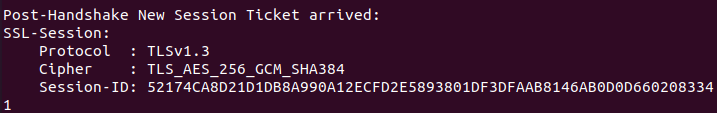
Во второй строке можно увидеть, что публичный ключ имеет длину 256 бит.

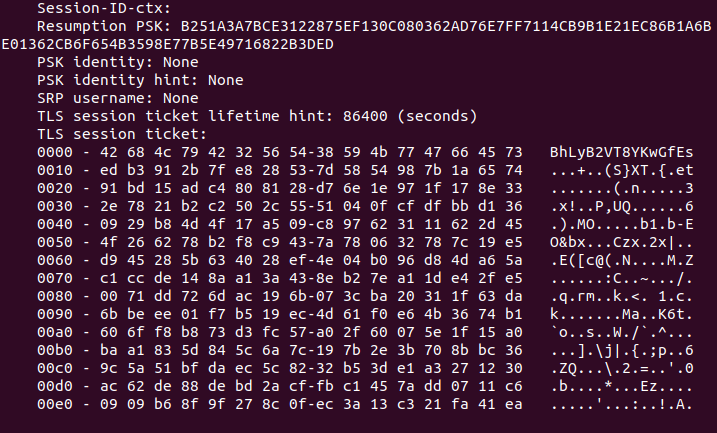
Третья строка означает, что не поддерживается возобновление TLS-сессии.

В четвертой строке указано, что сжатие не поддерживается.

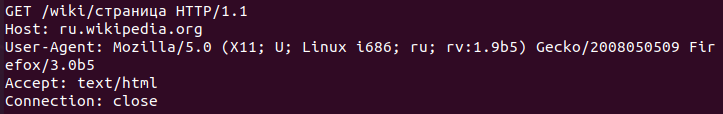
В пятой строке указано, что расширение не поддерживается.

Последняя строка указывает результат проверки.





Данные два блока означают, что данные текущей SSL/TLS сессии используются для обмена ключом для симметричного шифрования. Ключи меняются каждую новую сессию.



Отправляемый GET-запрос.

GET /wiki/страница HTTP/1.1 – это означает, что я отправляю GET запрос по протоколу HTTP версия которого 1.1 к ресурсу /wiki/страница

Host: ru.wikipedia.org – это хост, к которому отправляется запрос

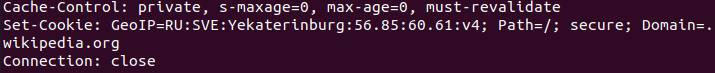
User-Agent: Mozila/5.0 (X11; U; Linux i686; ru; rv:1.9b5) Gecko/2008050509 Firefox/3.0b5 – это клиентское приложение, которое я использую

Accept: text/html – это означает, какой тип данных в ответе ожидается получить, а именно текстовый html-ответ

Connection: close – это закрытие соединения.

Ответ сервера: Изображение выглядит как текст

Автоматически созданное описание



В ответ я получил код состояния 301, находится в самой первой строке HTTP/1.1 301 Moved Permanently. Это значит, что ресурс перемещён навсегда. В Location находится ссылка на перемещённый ресурс.

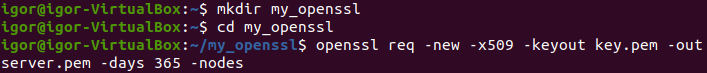
**Задание 2.**

Постановка задачи:

Изображение выглядит как текст

Автоматически созданное описание

Описание выполнения работы:



mkdir my\_openssl – это создание директории

cd my\_openssl – переход в эту директорию

В последней строке идёт создание ключей шифрования: публичного и приватного, их срок использования 365 дней.

Изображение выглядит как текст

Автоматически созданное описание

Заполнение информации в процессе генерации приватного ключа

Изображение выглядит как текст

Автоматически созданное описание

Созданные ключи находится в директории my\_openssl



Создаю файл форматом .py, файл будет находится в директории my\_openssl

Изображение выглядит как текст

Автоматически созданное описание

Указываю в файле следующее. Это необходимо для поднятия сервера.



Запускаю файл.



В новом терминале перехожу в директорию, где лежат созданные файлы. И отправляю запрос серверу.

Изображение выглядит как текст

Автоматически созданное описание

Вторая строка подтверждает факт установки соединения.

В остальных строках содержится результат проверки цепочки сертификатов, использованной при установке соединения. depth – это глубина каждого сертификата в цепочке сертификатов, С – страна, О – организация, L – город, ST – область, и verify return – результат проверки

Изображение выглядит как текст

Автоматически созданное описание

Здесь отображаются содержимое цепочки сертификатов, установленной на сервере (Certificate chain).

s – это назначение для каждого сертификата, которое содержит , С – страна, О – организация, СT – область, L - город.

i – это информация об издателе.

Изображение выглядит как текст

Автоматически созданное описание

Изображение выглядит как текст

Автоматически созданное описание

Выводимый текст сертификата сервера в формате PEM , который начинается с BEGIN и заканчивается END, его назначение subject и информация об издателе issuer.

Изображение выглядит как текст

Автоматически созданное описание

У нас самоподписанный сертификат SSL, поэтому выводятся такие команды. Тут определен формат подписи: SHA256 RSA-PSS

Изображение выглядит как текст

Автоматически созданное описание

В первой строке можно увидеть, что используется новый протокол TLSv1.3 и набор шифров Cipher is TLS\_AES\_256\_GCM\_SHA384. Это генерация сеансового ключа. Алгоритм AES256 – для шифрования трафика с длиной ключа 256 бит в режиме GCM, алгоритм SHA384 – для контроля целостности.

Во второй строке можно увидеть, что публичный ключ имеет длину 2048 бит.

Третья строка означает, что не поддерживается возобновление TLS-сессии.

В четвертой строке указано, что сжатие не поддерживается.

В пятой строке указано, что расширение не поддерживается.

Последняя строка указывает результат проверки

Изображение выглядит как текст

Автоматически созданное описание

Данные два блока означают, что данные текущей SSL/TLS сессии используются для обмена ключом для симметричного шифрования. Ключи меняются каждую новую сессию.

Отправим GET-запрос серверу.

Изображение выглядит как текст

Автоматически созданное описание

Ответ сервера:

Изображение выглядит как текст

Автоматически созданное описание

Текстовый-html ответ.

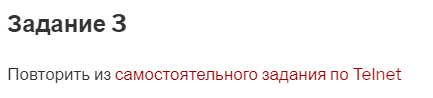
Изображение выглядит как знак, фотография, закрыть, синий

Автоматически созданное описание

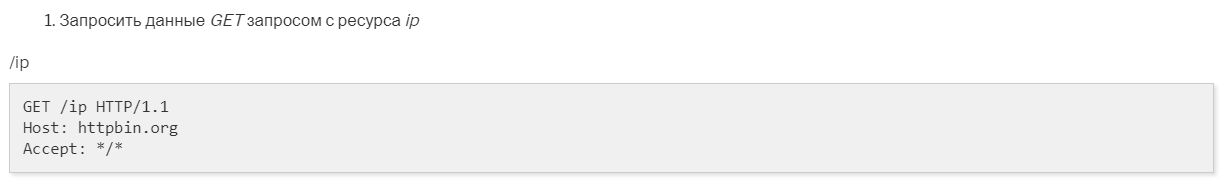
Показано, что сервер обработал мой запрос.

**Задание 3.**

Постановка общей задачи:



1) Постановка задачи первого задания:



Описание выполнения работы:

openssl s\_client –connect httpbin.org:443 - устанавливаем соединение с сервером, порт 443.

Отправляемый запрос:



GET /ip HTTP/1.1 – это означает, что я отправляю GET запрос по протоколу HTTP версия которого 1.1 для ресурса /ip

Host: httpbin.org – это хост, к которому отправляется запрос

Accept: \*/\* – это означает, что ожидается принятие любых типов данных

Ответ сервера:

Изображение выглядит как текст

Автоматически созданное описание

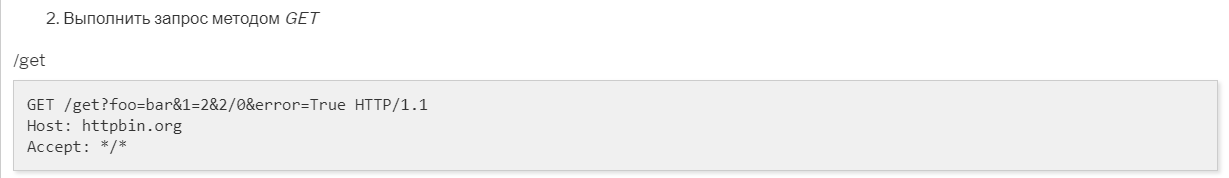
HTTP/1.1 200 OK – код состояния выполнения запроса, который означает, что выполнение прошло успешно

Content-Type: application/json – означает, что данные выводятся в формате json, которые предоставляют нам ip-адрес

Content-Length: 33 – длинна ответ 33 символа

“origin”: “178.46.122.149” – это тот ответ в формате json, длинною 33 символа

2) Постановка задачи:



Описание выполнения работы:

openssl s\_client –connect httpbin.org:443 - устанавливаем соединение с сервером, порт 443.

Отправляемый запрос:



GET /get?foo=bar&1=2&2/0&error=True HTTP/1.1 – это означает, что я отправляю GET запрос c использованием метода get, который получает набор аргументов, по протоколу HTTP версия которого 1.1

Host: httpbin.org – это хост, к которому отправляется запрос

Accept: \*/\* – это означает, что ожидается принятие любых типов данных

Ответ сервера:

Изображение выглядит как текст

Автоматически созданное описание

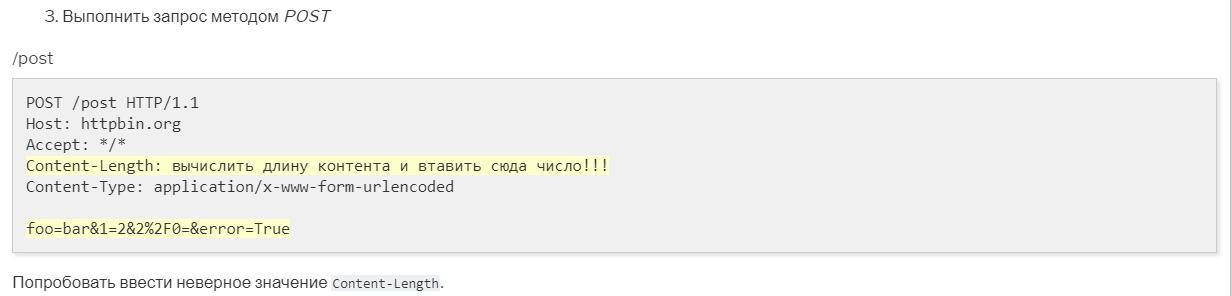
HTTP/1.1 200 OK – код состояния выполнения запроса, который означает, что выполнение прошло успешно

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

Content-Length: 324 – длинна ответ 324 символа

И были получены в результате json-аргументы, которые мы передали серверу методом get

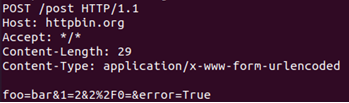
3) Постановка задачи:



Описание выполнения работы:

openssl s\_client –connect httpbin.org:443 - устанавливаем соединение с сервером, порт 443.

Отправляемый запрос:



POST /post HTTP/1.1 – это означает, что я отправляю POST запрос c использованием метода post, который записывает передаваемые данные, по протоколу HTTP версия которого 1.1

Host: httpbin.org – это хост, к которому отправляется запрос

Accept: \*/\* – это означает, что ожидается принятие любых типов данных

Content-Length: 29 – длина контента

Content-Type: application/x-form-urlencoded – это тип передаваемых данных

foo=bar&1=2&2%2F0=&error=True – контент длинною 29

Ответ сервера:

Изображение выглядит как текст

Автоматически созданное описание

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

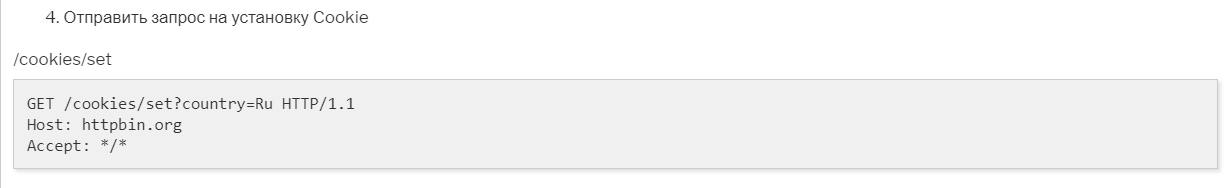
Content-Type: application/json – означает, что данные выводятся в формате json

Content-Length: 373 – длинна ответа 373 символа

Переданная информация была записана в поле form.

При вводе неверной длинны будем получать: код состояния ответа 400 Bad Request, что означает, что при передаче данных потерялась часть данных.

4) Постановка задачи:



Описание выполнения работы:

openssl s\_client –connect httpbin.org:443 - устанавливаем соединение с сервером, порт 443.

Отправляемый запрос:



Ответ сервера:

Изображение выглядит как текст

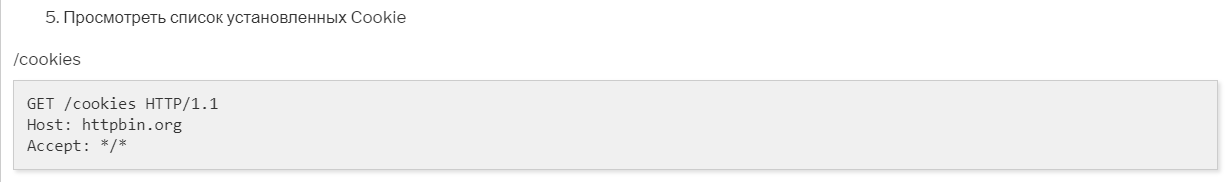
Автоматически созданное описание

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=/ - означает, что запрос был получен

Результатом служит перенаправление на другую страницу

5) Постановка задачи:



Описание выполнения работы:

openssl s\_client –connect httpbin.org:443 - устанавливаем соединение с сервером, порт 443.

Отправляемый запрос:



Ответ сервера:

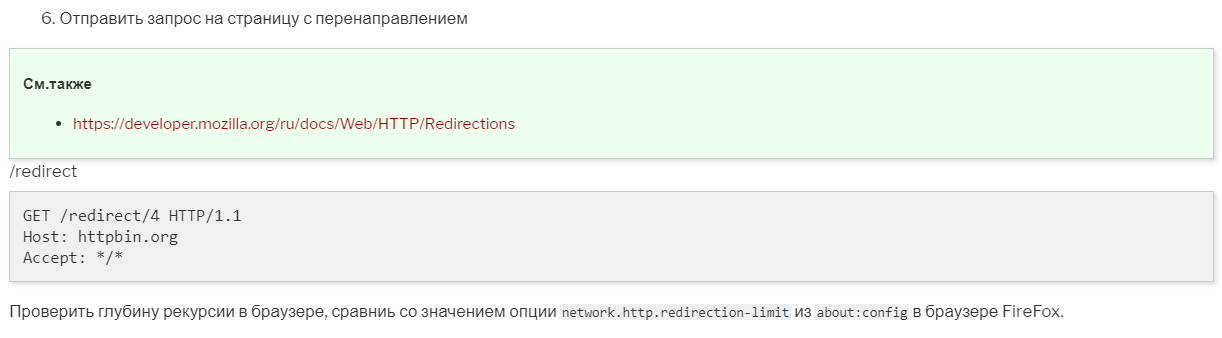
Изображение выглядит как текст

Автоматически созданное описание

Код состояния запроса 200, что значит запрос выполнен успешно.

“cookies”: {} – означает, что список установленных cookie пустой, так в прошлом шаге, установки не произошло, было перенаправление.

6) Постановка задачи:



Описание выполнения задания:

openssl s\_client –connect httpbin.org:443 - устанавливаем соединение с сервером, порт 443

Отправляемый запрос:



Ответ сервера:

Изображение выглядит как текст

Автоматически созданное описание

Код состояния выполнения запроса 404 Not Found. То есть указанный ресурс не найден.