Лабораторная работа № 6

Программирование безопасного клиент-серверного взаимодействия с использованием HTTPS

1. Теоретические сведения

HTTPS (аббр. от англ. *HyperText Transfer Protocol Secure*) — расширение протокола HTTP, поддерживающее шифрование. Данные, передаваемые по протоколу HTTPS, «упаковываются» в криптографический протокол SSL или TLS. В отличие от HTTP, для HTTPS по умолчанию используется TCP-порт 443.

Протокол был разработан компанией Netscape Communications для браузера Netscape Navigator в 1994 году[1]. HTTPS широко используется в мире веб и поддерживается всеми популярными браузерами.

HTTPS не является отдельным протоколом. Это обычный HTTP, работающий через шифрованные транспортные механизмы SSL и TLS. Он обеспечивает защиту от атак, основанных на прослушивании сетевого соединения — отснифферских атак и атак типа man-in-the-middle, при условии, что будут использоваться шифрующие средства и*сертификат сервера проверен и ему доверяют*.

По умолчанию HTTPS URL использует 443 TCP-порт (для незащищённого HTTP — 80). Чтобы подготовить веб-сервер для обработки https-соединений, администратор должен получить и установить в систему сертификат для этого веб-сервера. Сертификат состоит из 2 частей (2 ключей) — public и private. Public-часть сертификата используется для зашифровывания трафика от клиента к серверу в защищённом соединении, private-часть — для расшифровывания полученного от клиента зашифрованного трафика на сервере. После того как пара ключей приватный/публичный сгенерированы, на основе публичного ключа формируется запрос на сертификат вЦентр сертификации, в ответ на который ЦС высылает подписанный сертификат. ЦС при подписании проверяет клиента, что позволяет ему гарантировать, что держатель сертификата является тем, за кого себя выдаёт (обычно это платная услуга).

Существует возможность создать такой сертификат, не обращаясь в ЦС. Такие сертификаты могут быть созданы для серверов, работающих под Unix, с помощью таких утилит, как ssl-ca от OpenSSL или gensslcert от SuSE. Подписываются такие сертификаты этим же сертификатом и называются самоподписанными (*self-signed*). Без проверки сертификата каким-то другим способом (например, звонок владельцу и проверка контрольной суммы сертификата) такое использование HTTPS подвержено атаке man-in-the-middle.

Эта система также может использоваться для аутентификации клиента, чтобы обеспечить доступ к серверу только авторизованным пользователям. Для этого администратор обычно создаёт сертификаты для каждого пользователя и загружает их в браузер каждого пользователя. Также будут приниматься все сертификаты, подписанные организациями, которым доверяет сервер. Такой сертификат обычно содержит имя и адрес электронной почты авторизованного пользователя, которые

проверяются при каждом соединении, чтобы проверить личность пользователя без ввода пароля.

Создание сертификата

```
makecert
    -pe                                     Exportable private key
    -n "CN=Test"                            Subject name
    -ss my                                  Certificate store name
    -sr LocalMachine                        Certificate store location
    -a sha1                                 Signature algorithm
    -sky signature                          Subject key type is for signature
purposes
    -r                                      Make a self-signed cert
    "output.cer"
```

```
makecert -sv key.pvk -n "CN=localhost"   -sr LocalMachine     -a sha1
-sky signature -r  "output2.cer"
```

```
PVK2PFX -pvk key.pvk -spc output2.cer -pfx yourpfxout2.pfx -po 123
```

После чего нужно сертификат зарегистрировать в windows, для этого можно 2 раза кликнуть и следовать инструкциям

Пример приложения на C#

```
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Net.Security;
using System.Net.Sockets;
using System.Security.Authentication;
using System.Security.Cryptography.X509Certificates;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading;


namespace https
{
   class Program
   {
       public static string SuccessHeaders(int contentLength)
```

```
        {
            StringBuilder builder = new StringBuilder();
            builder.Append("HTTP/1.1 200 OK").Append("\r\n");
            builder.Append("Date: ").Append(DateTime.Now).Append("\r\n");
            builder.Append("Server: nesterione-server").Append("\r\n");
            builder.Append("Content-Type: text/html; charset=UTF-8").Append("\r\n");
            builder.Append("Content-Length: ").Append(contentLength).Append("\r\n");
            builder.Append("Connection: close").Append("\r\n");
            builder.Append("\r\n");

            return builder.ToString();
        }

        private static string AnswerPage(String val) {
            StringBuilder bodyBuilder = new StringBuilder();
            bodyBuilder.Append("You entered value:");
            bodyBuilder.Append("<br><br>");
            bodyBuilder.Append(val);
            bodyBuilder.Append("<br>");
            bodyBuilder.Append("<br>");
            bodyBuilder.Append("<a href='/' > Open Index Page </a> ");
            String body = bodyBuilder.ToString();

            // warning, if body contains non-ascii symbols next record can be
incorect
            return String.Concat(SuccessHeaders(body.Length), body);
        }

        private static string BadAnswerPage()
        {
            StringBuilder bodyBuilder = new StringBuilder();
            bodyBuilder.Append("Something was bad");
            bodyBuilder.Append("<br>");
            bodyBuilder.Append("<a href='/' > Open Index Page </a> ");
            String body = bodyBuilder.ToString();

            // warning, if body contains non-ascii symbols next record can be
incorect
```

```
            return String.Concat(SuccessHeaders(body.Length), body);
        }


        private static string IndexPage()
        {
            StringBuilder bodyBuilder = new StringBuilder();
            bodyBuilder.Append("<form method='post'>");
            bodyBuilder.Append("hi, I'm https-server, input number:");
            bodyBuilder.Append("<br>");
            bodyBuilder.Append("<input type='text' name='val' >");
            bodyBuilder.Append("<br>");
            bodyBuilder.Append("<input type='submit' >");
            bodyBuilder.Append("</form>");
            String body = bodyBuilder.ToString();


            // warning, if body contains non-ascii symbols next record can be
incorect
            return String.Concat(SuccessHeaders(body.Length), body);
        }



        static X509Certificate serverCertificate = null;


        static void ProcessClient(Object obj)
        {
            TcpClient client = (TcpClient)obj;


            // A client has connected. Create the
            // SslStream using the client's network stream.
            SslStream sslStream = new SslStream(client.GetStream(), false);
            // Authenticate the server but don't require the client to authenticate.
            try
            {
                sslStream.AuthenticateAsServer(serverCertificate,
                    false, SslProtocols.Tls, true);
                // Display the properties and settings for the authenticated stream.
```

```
            // Set timeouts for the read and write to 5 seconds.
            sslStream.ReadTimeout = 5000;
            sslStream.WriteTimeout = 5000;
            // Read a message from the client.
            Console.WriteLine("Waiting for client message...");

            string messageData = ReadMessage(sslStream);


            Console.WriteLine("request:");
            Console.WriteLine(messageData);


            string page = "";
            if (messageData.StartsWith("POST"))
            {
                Regex r = new Regex("\r\n\r\n");
                String[] request = r.Split(messageData, 2);


                if(request.Length!=2)
                {
                    page = BadAnswerPage();
                } else
                {
                    String val = request[1].Split('=')[1];
                    page = AnswerPage(val);
                }

            } else
            {
                page = IndexPage();
            }


            Console.WriteLine("response:");
            Console.WriteLine(page);
            byte[] message = Encoding.UTF8.GetBytes(page);
            sslStream.Write(message,0,message.Length);
            sslStream.Flush();
        }
        catch (AuthenticationException e)
        {
```

```
                Console.WriteLine("Exception: {0}", e.Message);
                if (e.InnerException != null)
                {
                    Console.WriteLine("Inner exception: {0}",
e.InnerException.Message);
                }
                Console.WriteLine("Authentication failed - closing the
connection.");
            }
            finally
            {
                // The client stream will be closed with the sslStream
                // because we specified this behavior when creating
                // the sslStream.
                sslStream.Close();
                client.Close();
            }
        }


        static void Main(string[] args)
        {
            serverCertificate = X509Certificate.CreateFromCertFile("output2.cer");
            TcpListener listener = new TcpListener(IPAddress.Any, 8088);
            listener.Start();
            Console.WriteLine("Server run on: https://localhost:8088 ");
            while (true)
            {
                TcpClient client = listener.AcceptTcpClient();
                ThreadPool.QueueUserWorkItem(ProcessClient, client);
            }
        }


        static string ReadMessage(SslStream sslStream)
        {
            StringBuilder messageData=new StringBuilder();
            try {
                byte[] buffer = new byte[2048];
                int bytes = -1;
                do
                {
                    // Read the client's test message.
```

```
                    bytes = sslStream.Read(buffer, 0, buffer.Length);


                    // Use Decoder class to convert from bytes to UTF8
                    // in case a character spans two buffers.
                    Decoder decoder = Encoding.UTF8.GetDecoder();
                    char[] chars = new char[decoder.GetCharCount(buffer, 0, bytes)];
                    decoder.GetChars(buffer, 0, bytes, chars, 0);
                    messageData.Append(chars);

                } while (bytes != 0);
            } catch(Exception) { }


            return messageData.ToString();
        }
    }
}
```

трасировка запросов при работе представленного выше приложения

Server run on: https://localhost:8088 Waiting for client message... request: GET / HTTP/1.1 Accept: text/html, application/xhtml+xml, image/jxr, / Accept-Language: en-US,en;q=0.7,ru;q=0.3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2486.0 Safari/537.36 Edge/13.10586 Accept-Encoding: gzip, deflate Host: localhost:8088 Connection: Keep-Alive

response: HTTP/1.1 200 OK Date: 26.11.2015 2:49:35 Server: nesterione-server Content-Type: text/html; charset=UTF-8 Content-Length: 123 Connection: close

hi, I'm https-server, input number:

Waiting for client message... Waiting for client message... request: GET / HTTP/1.1 Accept: text/html, application/xhtml+xml, image/jxr, / Accept-Language: en-US,en;q=0.7,ru;q=0.3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2486.0 Safari/537.36 Edge/13.10586 Accept-Encoding: gzip, deflate Host: localhost:8088 Connection: Keep-Alive

response: HTTP/1.1 200 OK Date: 26.11.2015 2:49:44 Server: nesterione-server Content-Type: text/html; charset=UTF-8 Content-Length: 123 Connection: close

hi, I'm https-server, input number:

request: POST / HTTP/1.1 Accept: text/html, application/xhtml+xml, image/jxr, / Referer: https://localhost:8088/ Accept-Language: en-US,en;q=0.7,ru;q=0.3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2486.0 Safari/537.36 Edge/13.10586 Content-Type: application/x-www-form-urlencoded Accept-Encoding: gzip, deflate Host: localhost:8088 Content-Length: 8 Connection: Keep-Alive Cache-Control: no-cache

val=5555 response: HTTP/1.1 200 OK Date: 26.11.2015 2:49:49 Server: nesterione-server Content-Type: text/html; charset=UTF-8 Content-Length: 73 Connection: close

You entered value:

5555

Open Index Page Waiting for client message... request: POST / HTTP/1.1 Accept: text/html, application/xhtml+xml, image/jxr, /Referer: https://localhost:8088/ Accept-Language: en-US,en;q=0.7,ru;q=0.3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/46.0.2486.0 Safari/537.36 Edge/13.10586 Content-Type: application/x-www-form-urlencoded Accept-Encoding: gzip, deflate Host: localhost:8088 Content-Length: 9 Connection: Keep-Alive Cache-Control: no-cache

val=hello response: HTTP/1.1 200 OK Date: 26.11.2015 2:49:54 Server: nesterione-server Content-Type: text/html; charset=UTF-8 Content-Length: 74 Connection: close

You entered value:

hello

2. Задание на лабораторную работу.

Для задания представленном ниже для коммуникации использовать HTTPS протокол (генерировать тестовый сертификат с использованием OpenSSL, можно использовать сертификаты генерируемые инструментами Windows). (Вся работа должна быть выполнена с использованием стандартных сокетов, все HTTPS заголовки формируются вручную)

Разработать https-сервер (использовать пул потоков для параллельной обработки запросов клиентов). В качестве клиентского приложения используется веб-браузер. Все заголовки формируются вручную. Браузер должен правильно отображать получаемую информацию. Рекомендуется на GET-запрос возвращать html страницу с формой ввода, и выполнять POST-запрос, на который сервер присылает решение.

Отчѐт должен содержать:

- Краткое описание алгоритма
- Описание используемых http заголовков
- Верификация

Варианты заданий

Передается только текстовая информация. Т.е. параметры задачи передаются в теле запроса, ответ приходит в виде html-страницы.	4-5 баллов
Задание передается на сервер в виде текста (вводится в поля ввода html-формы). В ответ возвращает прикрепленный файл, с результатом решения задачи. Формат файла разрешается любой, рекомендуется XML или простой текстовый файл.	6-8 баллов
Задание на сервер передается в виде файла, например изображения или текстового файла с параметрами задачи(в форме ввода используется input для отправки файла). В ответ сервер возвращает прикрепленный файл. Для заданий с фильтрами, это обработанное изображение, для остальных - текстовый файл в любом формате.	9-10 баллов

Варианты согласуются с преподавателем:

1.	Вычисление псевдослучайного числа используя регистр сдвига с линейной обратной связью (LFSR)
2	Одномерная оптимизация функции методом деления попалам
3	Mergesort

4	Одномерная оптимизация функции методом золотого сечения
5	Quicksort
6	Heapsort
7	Insertion Sort
8	Одномерная оптимизация функции методом равномерного поиска
9	Умножение матриц
10	Одномерная оптимизация функции методом Фибоначчи
11	Treesort
12	Одномерная оптимизация функции методом касательных
13	Разбор математического выражения и вычисление с учетом скобок и приоритетов операций (+ - * /) . Использовать обратную польскую запись
14	Вычисление секанса (без использования математических библиотек, используя разложение в ряд Маклоррена)
15	Вычисление синуса (без использования математических библиотек, используя разложение в ряд Маклоррена)
16	Вычисление экспоненты (без использования математических библиотек, используя разложение в ряд Маклоррена)
17	Разбор математического выражения с учетом приоритетов операций (+ - * /), и тригонометрических функций (sin, cos, tg) . Использовать обратную польскую запись.
18	Вычислить число ПИ до 50 знака методом Монте-Карло
19	Вычислить число ПИ до 50 знака методом иглы Буффона
20	Вычислить число ПИ до 50 знака разложением в ряд Нилаканта
21	Вычислить квадратный корень из числа, используя итерационную формулу Герона
22	Вычислить квадратный корень из числа, используя алгоритм Ньютона
23	Вычисление натурального логарифма (без использования математических библиотек, используя разложение в ряд Маклоррена)
24	Вычисление гиперболического синуса(без использования математических библиотек, используя разложение в ряд Маклоррена)
25	Вычислить число ПИ до 50 знака разложением в ряд Лейбница
26	Решение СЛАУ через обратную матрицу
27	Разработать конечный автомат для анализа вырожения.

	Входной язык содержит последовательность вызовов процедур, разделенных символом ;(точка с запятой). Вызов процедуры должен состоять из имени процедуры и списка параметров. В качестве параметров могут выступать идентификаторы, строковые константы, заключенные в двойные кавычки и одиночные символы, заключенные в одинарные кавычки. Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
28	Разработать конечный автомат для анализа вырожения. Входной язык содержит арифметические выражения, разделенные символом ;(точка с запятой). Арифметические выражения состоят из идентификаторов, римских чисел, знаков операций и скобок. (Римскими считать числа записанные большими буквами X, V и I). Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
29	Разработать конечный автомат для анализа вырожения. Входной язык содержит HTML код, нужно проверить правильность объявления ссылки. Учитывать что могут быть кавычки двух типов или отсутствовать. Проверка атрибутов href, alt, blank. Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
30	Разработать конечный автомат для анализа вырожения. Входной язык содержит HTML код, нужно проверить правильность тега img. Учитывать что могут быть кавычки двух типов или отсутствовать. Проверка атрибутов src, alt, width, height. Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
31	Разработать конечный автомат для анализа корректности email. Найти все email в файле отправленном пользователем. Клиенту возвращается список всех найденных email в тексте
32	Найти обратную матрицу (используя метод Гаусса)
33	Разбор математического выражения и вычисление с учетом скобок и приоритетов операций (+ - * /) . Использовать дерево выражений

34	Граф хранится на сервере, пользователь указывает название точки отправления и точки назначения, в ответ получает кратчайший путь. Для поиска пути использовать Алгоритм Флойда — Уоршелла
35	Граф хранится на сервере, пользователь указывает название точки отправления и точки назначения, в ответ получает кратчайший путь. Для поиска пути использовать Алгоритм Дейкстры
36	Граф хранится на сервере, пользователь указывает название точки отправления и точки назначения, в ответ получает кратчайший путь. Для поиска пути использовать Алгоритм Беллмана — Форда
37	Решение СЛАУ методом Гаусса
38	Решение СЛАУ методом Жардана-Гауса
39	Найти определитель матрицы
40	LDL^T разложение матрицы
41	Найти обратную матрицу (через единичную)
42	Разложение Холецкого
43	LU разложение матрицы
44	Умножение матриц алгоритмом Штрассена
45	Шифрование строки алгоритмом RSA
46	Разработать конечный автомат для анализа вырожения. Входной язык содержит последовательность описаний массивов в соответствии со спецификацией языка C#, разделенных символом ;(точка с запятой). Считать, что массивы могут содержать только элементы следующих типов: int, double, float, char и String. Массивы инициализируют только фиксированной длины. Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
47	Разработать конечный автомат для анализа вырожения. Входной язык содержит последовательность команд ассемблера в форме: <метка>: <команда> <операнд1>,<операнд2> (метка, а также один или оба операнда могут отсутствовать). В качестве операндов могут выступать регистры процессора 80x86, идентификаторы, целые десятичные числа или целые шестнадцатеричные числа. (Предусмотреть наличие не менее 6 допустимых команд). Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.

48	Разработать конечный автомат для анализа вырожения. Входной язык содержит последовательность описаний записей (record) в соответствии со спецификацией языка Паскаль, разделенных символом ;(точка с запятой). Считать, что записи могут содержать только поля скалярных типов integer, real, byte, word, char и строки string с возможным указанием длины строки в квадратных скобках. Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
49	Разработать конечный автомат для анализа вырожения. Входной язык содержит последовательность описаний массивов в соответствии со спецификацией языка Паскаль, разделенных символом ;(точка с запятой). Считать, что массивы могут содержать только элементы скалярных типов integer, real, byte, word и char. Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
50	Разработать конечный автомат для анализа вырожения. Входной язык содержит последовательность вызовов процедур, разделенных символом ;(точка с запятой). Вызов процедуры должен состоять из имени процедуры и списка параметров. В качестве параметров могут выступать идентификаторы, целые десятичные числа без знака, шестнадцатеричные числа, десятичные числа с плавающей точкой. Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
51	Разработать конечный автомат для анализа вырожения. Входной язык содержит выражения над строковыми константами, разделенные символом ;(точка с запятой). Выражения состоят из идентификаторов, строковых констант, заключенных в двойные кавычки, одиночных символов, заключенных в одинарные кавычки и знаков операции конкатенации +. Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
52	Разработать конечный автомат для анализа вырожения.

	Входной язык содержит упрощенные условные операторы типа if <логическое выражение> then <оператор присваивания> else <оператор присваивания>; (часть else в операторе может отсутствовать). Логическое выражение может содержать идентификаторы, знаки операций сравнения, целые десятичные числа без знака, скобки и логические операции and и not. Оператор присваивания должен состоять из двух идентификаторов, разделенных знаком присваивания. Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
53	Разработать конечный автомат для анализа вырожения. Входной язык содержит упрощенные операторы цикла типа while <логическое выражение> do <оператор присваивания>; Логическое выражение может содержать идентификаторы, знаки операций сравнения, целые десятичные числа без знака, скобки и логические операции and и or. Оператор присваивания должен состоять из идентификатора, знака присваивания и целой десятичной константы без знака. Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
54	Разработать конечный автомат для анализа вырожения. Входной язык содержит арифметические выражения, разделенные символом ;(точка с запятой). Арифметические выражения состоят из идентификаторов, десятичных чисел с плавающей точкой (в обычной и логарифмической форме), знаков операций и скобок. Клиенту возвращается удовлетворяет ли переданное выражение заданному языку или нет. Обработку запросов от клиентов реализовать с использованием пула потоков.
55	решения систем обыкновенных дифференциальных уравнений Рунге-Кутты 4-го порядка
56	решения систем обыкновенных дифференциальных уравнений методом Эйлера
57	Разработать конечный автомат для анализа корректности номера телефона в формате.[+XXX (YY) XXX-XX-XX] Найти все номера телефонов в файле отправленном пользователем. Клиенту возвращается список всех найденных номеров телефонов в тексте
58	Разработать конечный автомат для анализа надежности и корректности

	пароля Длина пароля не менее 6 символов. Разрешены только латинские буквы, цифры и знаки: пробел _ - . В пароле обязательно должны присутствовать знаки в верхнем и нижнем регистре и цифры. Пользователю возвращается информация о том, можно ли использовать указанный пароль
59	Медианный фильтр изображения с возможностью выбора размера окна
60	Фильтр Гаусса для изображений с возможностью выбора размера окна
61	Фильтр Собеля (оператор Собеля) для выявления границ объектов изображения
62	Перекрестный оператор Робертса для выявления границ объектов на изображении
63	Оператор Прюитт для выделения границ на изображении
64	Проверка вхождения двух человек в одну группу. Исходные данные, имена людей, которые добавлены в друзья в социальной сети. Пользователь вводит 2 имени, нужно определить связаны ли они через общих друзей. (Find Union)
65	Компрессия/Декомпрессия данных. Алгоритм Лемпеля — Зива — Велча (LZW)
66	Компрессия/Декомпрессия данных. алгоритмом Хаффмана
67	Многомерная оптимизация функции методом деформируемого многогранника (Нелдера-Мида)
68	Многомерная оптимизация функции методом Хука Дживса
69	Многомерная оптимизация функции методом градиентного спуска (построить график)
70	Оператор Кенни для поиска границ объектов на изображении

3.Контрольные вопросы

1. Протокол HTTP
2. Виды запросов HTTP протокола
3. GET – запрос
4. POST – запрос
5. HEAD – запрос
6. PUT – запрос
7. TRACE – запрос
8. Отличие HTTPS протокола от HTTP
9. Основные коды ответов HTTP протокола

10. Основные процедуры и классы, для реализации сетевого взаимодействия