УДК 004.522

А.А. МЯСИЩЕВ
Хмельницкий национальный университет

АВТОРИЗАЦИЯ НА WEB-СЕРВЕРЕ ARDUINO ДЛЯ УПРАВЛЕНИЯ ЧЕРЕЗ ИНТЕРНЕТ

Рассматривается возможность построения web-сервера для управления оборудованием и получением данных с различных датчиков через сеть Интернет. Указывается на то, что для удаленного управления необходимо проводить авторизацию. В противном случае оборудование будет не защищено от несанкционированного доступа пользователей сети Интернет. Отмечается, что управляющие web-сервера создаются на базе микроконтроллеров, которые имеют малые ресурсы и не в состоянии работать с протоколами HTTPS, SSL, TLS. Поэтому эти сервера являются уязвимыми со стороны сетевых атак. В работе рассмотрено создание web-сервера на Arduino, который использует авторизацию на основе GET и POST запросов, а также модернизированную HTTP basic authentication. Модернизация состоит в том, что для авторизации используется пароль из списка паролей, который выбирается пользователем на основании ключа, пересылаемого сервером. При каждом новом входе на сервер предыдущий пароль становиться недействительным. Представлен практический пример web-сервера на Arduino Mega, на котором установлены три светодиода, имитирующие включение-выключение 3-х силовых источников питания (например, электра-розеток), датчик температуры DS18B20, датчик влажности и температуры DHT 11 и барометрический датчик BMP280. Сервер тестировался с тремя контроллерами Ethernet: enc28j60, W5100 и w5500. Для этого использовались три библиотеки: UIPEthetnet, Ethernet и Ethernet2, которые показали одинаковые результаты работы. Установлено, что авторизация на сервере Arduino с использованием метода запроса GET является нецелесообразной, так как при использовании форм ввода с полями для пароля коды открыто высвечиваются в адресной строке. Авторизация с использованием метода POST скрывает передаваемые коды. Но коды передаются по сети в открытой форме, поэтому их можно перехватить с помощью программ снифферов. Использование протоколов HTTPS, SSL, TLS позволяет сделать такую авторизацию безопасной. Программа, реализующая web-сервер Arduino, составлена в среде разработки Arduino IDE ver. 1.8.6.

Ключевые слова: Arduino, Ethernet Shield Arduino, ENC28J60, W5100, W5500, протокол HTTPS, SSL, TLS, GET и POST запросы, base64-encoded, basic authentication, web-server, микроконтроллер.

A.A. MYASISCHEV
Khmelnytsky National University

AUTHORIZATION ON WEB SERVER ARDUINO FOR CONTROLLING THROUGH THE INTERNET

The possibility of building a web server for controlling equipment and obtaining data from various sensors via the Internet is considered. It is indicated that authorization is necessary for remote management. Otherwise, the equipment will not be protected from unauthorized access of Internet users. It is noted that the managing web servers are created on the basis of microcontrollers, which have small resources and are not able to work with the protocols HTTPS, SSL, TLS. Therefore, these servers are vulnerable to network attacks. The paper considers the creation of a web server on Arduino, which uses authorization based on GET and POST requests, as well as modernized HTTP basic authentication. The upgrade consists in the fact that the password is used for authorization from the password list, which is selected by the user based on the key sent by the server. With each new login to the server, the previous password becomes invalid. A practical example of a web server is presented on the Arduino Mega, on which are installed three LEDs simulating switching on / off of 3 power supply sources (for example, power outlets), DS18B20 temperature sensor, humidity and temperature sensor DHT 11 and barometric sensor BMP280. The server was tested with three Ethernet controllers: enc28j60, W5100 and w5500. For this purpose, three libraries were used: UIPEthetnet, Ethernet and Ethernet2, which showed the same results. It has been established that authorization on the Arduino server using the GET request method is inappropriate, because when using input forms with password fields, the codes are displayed in the address bar. Authorization using the POST method hides the transmitted codes. But the codes are transmitted over the network in an open form, so they can be intercepted using snuffers programs. Using HTTPS, SSL, TLS allows you to make such authorization secure. The program that implements the Arduino web server is made in the Arduino IDE 1.8.6.

Keywords: Arduino, Ethernet Shield Arduino, ENC28J60, W5100, W5500, HTTPS, SSL, TLS, GET and POST requests, base64-encoded, basic authentication, web-server, microcontroller.

Постановка задачи

С появлением технологий удаленного управления оборудованием через сеть Интернет актуальной стала задача авторизации на управляющих серверах. Часто такими серверами являются web-сервера, построенные на основе микроконтроллеров среди которых наиболее распространены находящиеся в составе контроллеров Arduino. Контроллеры Arduino наиболее доступные, дешевые, имеют большое количество плат расширения (shield), бесплатную программную поддержку с большим количеством библиотек. Однако в отличие от современных компьютеров Arduino имеют скромные вычислительные ресурсы в связи с чем не в состоянии поддерживать полноценные сетевые протоколы, не говоря уже о программном обеспечении обеспечивающих шифрование данных в пакетах. Поэтому команды по управлению оборудованием посылаются с браузера клиента на web-сервер Arduino без пароля идентификации. Таким образом удаленно может управлять устройствами любой пользователь, которому известен адрес сервера. Это может нанести вред правильному функционированию систем, которые могут располагаться в офисе, предприятии, учреждении, жилом доме.

Для надежной авторизации на серверах с популярными сетевыми операционными системами (Linux, FreeBSD, OpenBSD, Windows Server...) используется протокол https, данные которого

«упаковываются» в криптографический протокол SSL или TLS, обеспечивая защиту этих данных. Работа этих протоколов выполняется с помощью специального программного обеспечения, которое не может быть установлено на сервера Arduino вследствие, в частности, малой памяти микроконтроллеров. Также и стек коммуникационных протоколов Ethernet Shield Arduino использует сокращенные, урезанные версии протоколов TCP/IP. Несмотря на указанные проблемы, рассмотрим возможность упрощенной удаленной авторизации на сервере Arduino. Для этого воспользуемся технологиями GET и POST запросов, а также HTTP authentication, описанной в стандартах HTTP 1.0/1.1.

Изложение основного материала работы

Рассмотрим web-сервер на Arduino, который управляет удаленно тремя светодиодами(имитируют работу удаленного оборудования) и получает данные с температурных датчиков и датчика влажности DHT11(или датчика давления BMP280). На рисунке 1 показан сервер на Arduino Mega с контроллером сети на ENC28J60 (или W5500), светодиодами ,релейным модулем для управления нагрузкой и LCD индикатором.

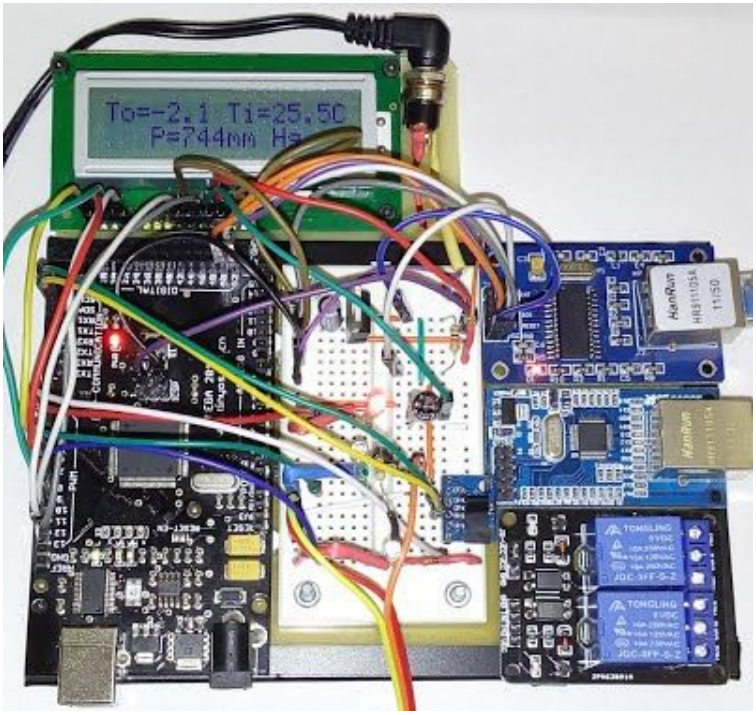

Рис. 1. Макет исследуемого web-сервера на Arduino Mega

Рассмотрим для авторизации доступа на аервер Ардуино технологии GET и POST запросов. Сервер на браузере клиента формирует форму, которая не должна отображать вводимые символы. Включение и выключение светодиодов выполняется вводом следующих кодов:

Коды управления светодиодами:
Включить красный - 11361
Выключить красный - 11360
Включить зеленый - 11381
Выключить зеленый - 11380
Включить синий - 11401
Выключить синий - 11400

Задача заключается в том, чтобы не удалось подсмотреть другому пользователю вводимые в форму коды управления. Шифрование данных по сети не рассматривается.

Рассмотрим GET запрос. Заголовок, посылаемый от браузера клиента к серверу имеет вид:

GET / HTTP/1.1
*Accept: text/html, application/xhtml+xml, image/jxr, */**
Accept-Language: ru,en-US;q=0.8,en;q=0.5,uk;q=0.3
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; MALNJS; rv:11.0) like Gecko
Accept-Encoding: gzip, deflate
Host: 192.168.1.18:81
Connection: Keep-Alive

В конце каждой строки заголовка следуют символы \r\n. В случае передачи параметров для включения красного и синего светодиодов заголовок будет следующим [1]:

GET /?ron=11361&roff=&gon=&goff=&bon=11401&boff= HTTP/1.1
*Accept: text/html, application/xhtml+xml, image/jxr, */**
Referer: http://192.168.1.18:81/

Accept-Language: ru,en-US;q=0.8,en;q=0.5,uk;q=0.3
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; MALNJS; rv:11.0) like Gecko
Accept-Encoding: gzip, deflate
Host: 192.168.1.18:81
Connection: Keep-Alive

Таким образом при GET запросе передача параметров на управление передается в первой строке заголовка браузера. На рисунке 2 представлен интерфейс управления. Здесь видно, что в адресной строке в открытом виде передаются все кодовые параметры на управление устройствами. Поэтому коды на управление можно легко подсмотреть с адресной строки. Программа сервера представлена в источнике [1].

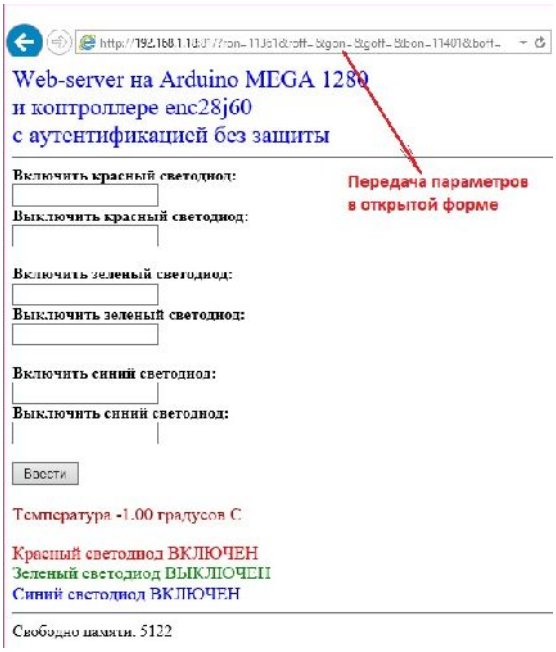

Рис. 2. Интерфейс управления для GET запроса

Рассмотрим POST запрос. В этом случае параметры управления(коды) передаются не через адресную строку. Заголовок, который посылает браузер при POST запросе, имеет следующий вид:

POST / HTTP/1.1
*Accept: text/html, application/xhtml+xml, image/jxr, */**
Referer: http://192.168.1.18:81/
Accept-Language: ru,en-US;q=0.8,en;q=0.5,uk;q=0.3
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64; Trident/7.0; MALNJS; rv:11.0) like Gecko
Accept-Encoding: gzip, deflate
Host: 192.168.1.18:81
Content-Length: 42
Connection: Keep-Alive
Cache-Control: no-cache
<---здесь пустая строка--->
ron=11361&roff=&gon=&goff=&bon=11401&boff=

Параметры для включения красного и синего светодиодов передаются после основного заголовка через пустую строку. Первая строка заголовка указывает на передачу POST запроса.

На рис. 3 представлен интерфейс управления для POST запроса.

Видно, что в адресной строке не передаются кодовые параметры на управление устройствами, например, при включении красного и синего светодиодов. Программа сервера для POST запроса представлена в источнике [2]. Для считывания кодового параметра используются операторы

```
while(client.available())   //Обработка запроса POST
      {   post = client.read();
        if (buffer.length() <= bufferMax) { buffer += post; } }
```

которые считывают часть заголовка, находящегося после пустой строки основной части заголовка. В массив buffer записываются управляющие параметры (ron=11361&roff=&gon=&goff=&bon=11401&boff=). В зависимости от их значения операторами

```
if(buffer.indexOf("ron=11361") >= 0) {digitalWrite(3, HIGH);}
if(buffer.indexOf("roff=11360") >= 0) {digitalWrite(3, LOW);}
if(buffer.indexOf("gon=11381") >= 0) {digitalWrite(5, HIGH);}
if(buffer.indexOf("goff=11380") >= 0) {digitalWrite(5, LOW);}
```

if(buffer.indexOf("bon=11401") >= 0) {digitalWrite(7, HIGH);}
if(buffer.indexOf("boff=11400") >= 0) {digitalWrite(7, LOW);}
выполняются соответствующие действия(включение или выключение).

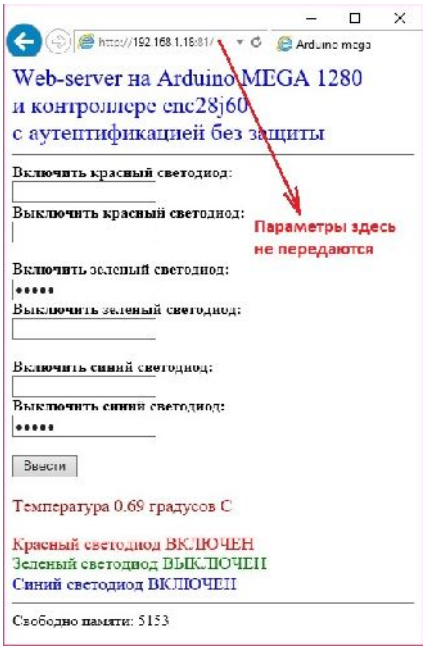

Рис. 3. Интерфейс управления для POST запроса

Из изложенного выше видно, что при POST запросе при вводе в форму кодов управления их значения подсмотреть невозможно(как и при вводе пароля при идентификации в системе через консоль). Однако данные передаются по сети в открытой форме, без кодирования. Поэтому используя программы sniffer можно всегда перехватить переданные данные и выявить управляющие коды. Использование протокола https (ssl, tsl) позволит передавать вводимые коды по сети с достаточной степенью безопасности.

Рассмотрим HTTP authentication. Этот протокол, описанный в стандартах HTTP 1.0/1.1, активно применяется в корпоративной среде. Применительно к web-сайтам он работает следующим образом:

1. Сервер, при обращении неавторизованного клиента к защищенному ресурсу, отсылает HTTP статус “401 Unauthorized” и добавляет заголовок “WWW-Authenticate” с указанием схемы и параметров аутентификации.

2. Браузер, при получении такого ответа, автоматически показывает диалог ввода username и password. Пользователь вводит детали своей учетной записи.

3. Сервер аутентифицирует пользователя по данным из этого заголовка.

Весь процесс стандартизирован поддерживается всеми браузерами и web-серверами. Существует несколько схем аутентификации, отличающихся по уровню безопасности. Ниже перечислены некоторые из них:

1. Basic – здесь username и password пользователя передаются в заголовке Authorization в незашифрованном виде (base64-encoded). Однако при использовании HTTPS (HTTP over SSL) протокола, basic является относительно безопасной (рис. 4).

Рис. 4. Схема Basic аутентификации

2. Digest — challenge-response-схема, при которой сервер посылает уникальное значение nonce, а браузер передает MD5 хэш пароля пользователя, вычисленный с использованием указанного nonce. Более безопасная альтернатив Basic схемы при незащищенных соединениях, но подвержена man-in-the-middle attacks (с заменой схемы на basic). Кроме того, использование этой схемы не позволяет применить современные хэш-функции для хранения паролей пользователей на сервере.

3. NTLM (известная как Windows authentication) — также основана на challenge-response подходе, при котором пароль не передается в чистом виде. Эта схема не является стандартом HTTP, но поддерживается большинством браузеров и веб-серверов. Преимущественно используется для аутентификации пользователей Windows Active Directory в веб-приложениях. Уязвима к pass-the-hash-атакам.

Рассмотрим схему аутентификации basic, и впоследствии модифицируем ее для веб-сервера Arduino. Согласно рис.4 практически механизм авторизации выглядит следующим образом. Сначала при запросе на подключение к серверу Arduino браузер посылает заголовок без запроса авторизации:

GET / HTTP/1.1
Host: 192.168.1.18:81
Connection: keep-alive
Cache-Control: max-age=0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,/*;q=0.8*
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/44.4.2403.3 Amigo/44.4.2403.3 MRCHROME SOC Safari/537.36
Accept-Encoding: gzip, deflate, sdch
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.6,en;q=0.4

Сервер Ардуино отвечает браузеру заголовком на необходимость запроса с авторизацией:

HTTP/1.0 401 Unauthorized
WWW-Authenticate: Basic realm="Arduino - HOME"

Браузер Amigo посылает серверу запрос, но уже с авторизацией:

GET / HTTP/1.1
Host: 192.168.1.18:81
Connection: keep-alive
Cache-Control: max-age=0
Authorization: Basic ***YWxleDoxMzY=***
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,/*;q=0.8*
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; WOW64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/44.4.2403.3 Amigo/44.4.2403.3 MRCHROME SOC Safari/537.36
Accept-Encoding: gzip, deflate, sdch
Accept-Language: ru-RU,ru;q=0.8,en-US;q=0.6,en;q=0.4
Сервер Arduino посылает ответ и после пустой строки html документ
HTTP/1.0 200 OK
Content-Type: text/html
<Здесь идет передача браузеру html документа>

YWxleDoxMzY= – это закодированный в Base64 набор символов alex:136 (alex – это login, 136 – password, которые вводятся в выскакивающем у браузере окне авторизации). На рисунке 5 представлены интерфейсы ввода логина и пароля и ответа сервера Arduino. Программа сервера Ардуино с описанной авторизацией представлена в источнике [3]. В этой программе работа сервера по представленной схеме реализуется с помощью операторов:

```
if (readString.lastIndexOf("YWxleDoxMzY=")>-1) {
  if (readString.lastIndexOf("GET /favicon.ico")>-1) {
    client.println("HTTP/1.0 404 Not Found");
  } else html_doc(client); }
else { client.println("HTTP/1.0 401 Unauthorized");
  client.println("WWW-Authenticate: Basic realm=\"Arduino - HOME\"");}
```

Однако безопасность при этой схеме весьма низкая. Пакеты по Интернет передаются в открытом виде, поэтому закодированный логин и пароль (YWxleDoxMzY=) можно легко выловить с помощью программ снифферов и раскодировать с помощью Base64 decode. Модернизируем программу, чтобы в большей степени защитить сервер Ардуино для предотвращения несанкционированного управления им. Для этого создадим список паролей, которые известны пользователю сервера. Причем при каждом новом входе на сервер пароль доступа должен меняться на следующий по списку. Поэтому, если предыдущий пароль будет расшифрован, то он становиться недействительный и злоумышленник не сможет зайти по нему. В источнике [3] представлена программа web-сервера Ардуино, на котором установлены три светодиода, имитирующие включение-выключение 3-х силовых источников питания (например электро-розеток), датчик температуры DS18B20(температура улицы) и датчик влажности и температуры DHT 11 (делает измерения в помещении) или датчик давления и температуры BMP280. Программа использует 3 пары логин: пароль:

1. alex:136; 2. alex:138; 3. alex:140

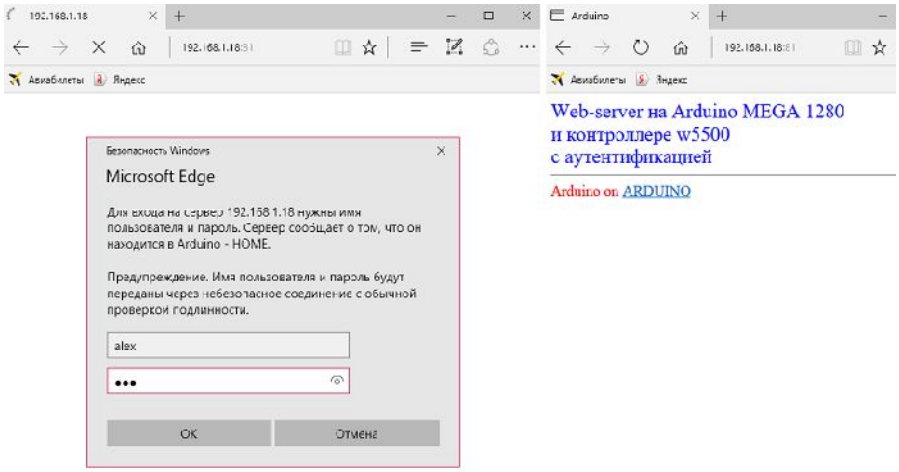

Рис. 5. Интерфейса ввода логина и пароля и ответа сервера Arduino

При каждом новом обращении к серверу пароль циклически меняется на следующий в списке. Для того, чтобы знать следующий номер пароля в браузере выводится параметр index. Его пользователь должен запомнить или определить с помощью открытого входа по адресу http://192.168.1.18:81/index. Таким образом, чем больше паролей в списке, тем более защищенным должен выглядеть сервер. На рисунке 6 представлен интерфейс управления сервером на Arduino Mega.

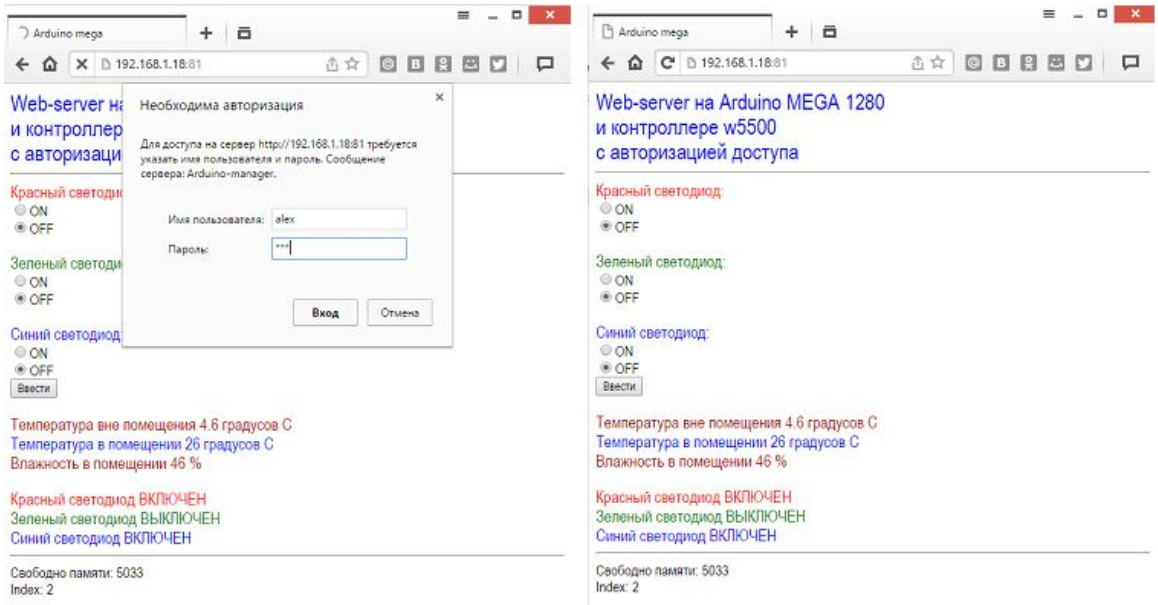

Рис. 6. Интерфейс управления web-сервером на Arduino Mega

В представленном интерфейсе Index:2 обозначает то, что при следующем входе на сервер необходимо для авторизации использовать пару alex:138

В программе работа сервера по представленной схеме для нескольких изменяющихся паролей реализуется с помощью операторов:

```
// 1-й пароль
      else if (readString.lastIndexOf("YWxleDoxMzY=")>-1 && passwd==1) {
        passwd=2; if (readString.lastIndexOf("GET /favicon.ico")>-1) {
          client.println("HTTP/1.0 404 Not Found"); }
        else { onoff(buffer); html_doc(client);} }
// 2-й пароль
       else if (readString.lastIndexOf("YWxleDoxMzg=")>-1 && passwd==2) {
        passwd=3; if (readString.lastIndexOf("GET /favicon.ico")>-1) {
          client.println("HTTP/1.0 404 Not Found"); }
        else {onoff(buffer); html_doc(client);} }
 // 3-й пароль  ...
```

Передача данных выполняется с помощью POST запроса. Данные запроса размещаются после пустой строки заголовка, который посылает браузер web-серверу Arduino:

```
POST / HTTP/1.1
Accept: text/html, application/xhtml+xml, image/jxr, */*
```

Referer: http://192.168.1.18:81/
Accept-Language: ru,en-US;q=0.8,en;q=0.5,uk;q=0.3
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.79 Safari/537.36 Edge/14.14393
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
Host: 192.168.1.18:81
Content-Length: 23
Connection: Keep-Alive
Cache-Control: no-cache
Authorization: Basic YWxleDoxMzY=
< пустая строка >
r=0&g=1&b=0&av=2018year

Последняя строка – это передача данных методом POST. Эти данные должны быть обработаны сервером для управления светодиодами (используется форма с радио кнопками).

Включение и выключение светодиодов определяется параметрами r, g и b. Если r=1 – красный светодиод включен, если r=0 – выключен. Аналогично для зеленого(g) и синего(b). Это выполняется операторами:

```
if(buffer.indexOf("r=1") >= 0) {digitalWrite(3, HIGH);}
if(buffer.indexOf("r=0") >= 0) {digitalWrite(3, LOW);}
if(buffer.indexOf("g=1") >= 0) {digitalWrite(5, HIGH);}
if(buffer.indexOf("g=0") >= 0) {digitalWrite(5, LOW);}
if(buffer.indexOf("b=1") >= 0) {digitalWrite(7, HIGH);}
if(buffer.indexOf("b=0") >= 0) {digitalWrite(7, LOW);}
```

После авторизированного входа на страничку управления работу с сервером удобно организовать так, чтобы после каждого изменения параметров светодиодов вход был не авторизированным. Для этого форма должна посылать серверу какой-то дополнительный код(набор символов), получив который сервер бы не запрашивал авторизацию. В этой программе посылается код 2018year с помощью оператора

```
client.println("<input type='hidden' name='av' value='2018year'>");
```

Так как поле имеет тип hidden, браузер его не отобразит, но методом PUT будет передано значение *2018year* с параметром *av (r=0&g=1&b=0&av=2018year).* С помощью операторов

```
else if (buffer.lastIndexOf("2018year")>-1 ) {
  if (readString.lastIndexOf("GET /favicon.ico")>-1) {
    client.println("HTTP/1.0 404 Not Found");  }
   else { onoff(); html_doc(client); } }
```

будет выполняться беспарольный вход на страничку управления. Однако для обеспечения лучшей защиты сервера Arduino из программы указанные выше операторы целесообразно убрать.

Выводы

1. Авторизация на сервере Arduino с использованием метода запроса GET является нецелесообразной, так как при использовании форм ввода с полями для пароля коды открыто высвечиваются в адресной строке. Поэтому рядом находящийся пользователь их легко может подсмотреть.

2. Авторизация с использованием метода POST скрывает передаваемые коды, однако коды передаются по сети в открытой, в не кодированной форме, поэтому их можно перехватить с помощью программ снифферов. Использование протокола HTTPS позволяет сделать такую авторизацию безопасной.

3. HTTP авторизация Basic передает username и password пользователя в незашифрованном виде (base64-encoded). И если не используется протокол HTTPS (HTTP over SSL), является относительно небезопасной.

4. Представленная в работе модернизация схемы авторизации Basic позволяет относительно безопасно управлять сервером Ардуино, но требует использования нескольких паролей, каждый из которых сообщается сервером пользователю через специальный параметр index.

5. В среде разработки Arduino IDE 1.8.6 разработано программное обеспечение, реализующее web-сервер на Arduino Mega для управления удаленными объектами через Интернет. Работоспособность опробована на макете, представленного на рис. 1.

Литература

1. Мясищев А.А. Web-server на Arduino MEGA и контроллере сети w5500 с авторизацией доступа для удаленного управления [Електронний ресурс]. – Режим доступу : https://sites.google.com/site/webstm32/web-server-avtorizaciej, 2018.

2. Мясищев А.А. GET и POST аутентификация на web-сервере Arduino [Електронний ресурс]. – Режим доступу : https://sites.google.com/site/webstm32/get-post-arduino, 2018.

3. Мясищев А.А. Web-server на Arduino mega с простой аутентификацией, использующей POST запрос [Електронний ресурс]. – Режим доступу : https://sites.google.com/site/webstm32/web-server--post-auten,

2018.

4. Kitsum. Авторизация на Web сервере микроконтроллера [Електронний ресурс]. – Режим доступу : https://it4it.club/topic/13-авторизация-на-web-сервере-микроконтроллера/, 2015

5. Гранкин С.С. Подключение Arduino к Интернету: настройка режима клиент-сервер, обработка GET и POST запросов [Електронний ресурс]. – Режим доступу : http://cxem.net/arduino/arduino176.php , 2016.

6. Выростков Д. Обзор способов и протоколов аутентификации в веб-приложениях / DataArt Технологический консалтинг и разработка ПО [Електронний ресурс]. – Режим доступу : https://habrahabr.ru/company/dataart/blog/262817, 2015.

7. Иго Т. Arduino, датчики и сети для связи устройств / Иго Т. ; пер. с англ. – 2-е изд. – СПб : БХВ-Петербург, 2015. – 544 с.

8. Иванов П.И. Управление блоком реле через браузер с помощью Arduino и Ethernet shield [Електронний ресурс]. – Режим доступу : http://www.psub.net/Publication/Details/60, 2015.

9. Скляров Д.В. Искусство защиты и взлома информации / Скляров Д.В. – СПб : БХВ-Петербург, 2004. – 288 с.

10. Анисимов В.В. Криптографические методы защиты информации [Електронний ресурс]. – Режим доступу : https://sites.google.com/site/anisimovkhv/learning/kripto/lecture, 2012.

11. Малков А. Классификация механизмов аутентификации пользователей и их обзор [Електронний ресурс]. – Режим доступу : https://habrahabr.ru/post/177551/, 2013.

12. Гавриков В. Шифрование данных: криптозащита STM32 [Електронний ресурс]. – Режим доступу : https://www.compel.ru/lib/ne/2016/10/2-shifrovanie-dannyih-kriptozashhita-stm32, 2016.

13. Шумель В.В., Рудикова Л.В. Общие принципы организации http-сервера на микроконтроллере [Електронний ресурс]. – Режим доступу : http://www.elib.bsu.by/bitstream/123456789/52523/1/41-45.pdf, 2013.

References

1. Myasishchev A.A. Web-server na Arduino MEGA i kontrollere seti w5500 s avtorizaciej dostupa dlya udalennogo upravleniya. [Electronic resource]. - Mode of access: https://sites.google.com/site/webstm32/web-server-avtorizaciej, 2018.

2. Myasishchev A.A. GET i POST autentifikaciya na web-servere Arduino. [Electronic resource]. - Mode of access: https://sites.google.com/site/webstm32/get-post-arduino, 2018.

3. Myasishchev A.A. Web - server na Arduino mega s prostoj autentifikaciej, ispol'zuyushchej POST zapros. [Electronic resource]. - Mode of access: https://sites.google.com/site/webstm32/web-server--post-auten, 2018.

4. Kitsum. Avtorizaciya na Web servere mikrokontrollera. [Electronic resource]. - Mode of access: https://it4it.club/topic/13-avtorizaciya-na-web-servere-mikrokontrollera/, 2015

5. Grankin S.S. Podklyuchenie Arduino k Internetu: nastrojka rezhima klient-server, obrabotka GET i POST zaprosov. [Electronic resource]. - Mode of access: http://cxem.net/arduino/arduino176.php , 2016.

6. Vyrostkov D. Obzor sposobov i protokolov autentifikacii v veb-prilozheniyah. / DataArt Tekhnologicheskij konsalting i razrabotka PO. - [Electronic resource]. - Mode of access: https://habrahabr.ru/company/dataart/blog/262817, 2015.

7. Igo.T. Arduino, datchiki i seti dlya svyazi ustrojstv: Per. s angl.- 2-e izd.- SPb.: BHV-Peterburg, 2015.- 544 s. il.

8. Ivanov P.I. Upravlenie blokom rele cherez brauzer s pomoshch'yu Arduino i Ethernet shield. [Electronic resource]. - Mode of access: http://www.psub.net/Publication/Details/60, 2015.

9. Sklyarov D.V. Iskusstvo zashchity i vzloma informacii. - SPb.: BHV-Peterburg, 2004.- 288 s. il.

10. Anisimov V.V. Kriptograficheskie metody zashchity informacii. [Electronic resource]. - Mode of access: https://sites.google.com/site/anisimovkhv/learning/kripto/lecture, 2012.

11. Malkov A. Klassifikaciya mekhanizmov autentifikacii pol'zovatelej i ih obzor. [Electronic resource]. - Mode of access: https://habrahabr.ru/post/177551/, 2013.

12. Gavrikov V. SHifrovanie dannyh: kriptozashchita STM32. [Electronic resource]. - Mode of access: https://www.compel.ru/lib/ne/2016/10/2-shifrovanie-dannyih-kriptozashhita-stm32, 2016.

13. SHumel' V.V., Rudikova L.V. Obshchie principy organizacii http-servera na mikrokontrollere. [Electronic resource]. - Mode of access: http://www.elib.bsu.by/bitstream/123456789/52523/1/41-45.pdf, 2013.

Рецензія/Peer review : 28.02.2018 р. Надрукована/Printed :23.03.2018 р.
Стаття прорецензована редакційною колегією