# 第六章：使用TCP/IP Sockets建立（安全）通信

## 目标

阅读第6章之后，您将理解如何使用WICED-SDK，运用TCP/IP Sockets发送和接收数据。您还将掌握对称和非对称加密的基本原理，以及如何使用加密为您的物联网设备提供安全性。

## 时间：2小时

## 基本原理

### Sockets – TCP通信基本原理

当应用（如网络浏览器）通过TCP传输层通信时，它们需要开通Sockets。Sockets，更严格来讲是TCP Sockets，简而言之就是互联网上两个设备间一条可靠、有序的通道。要开通Sockets，您需要在您准备通信的服务器上指定IP地址和端口号（一个无符号16位整数）。服务器上会运行一个程序，负责监听端口上流入的字节。Sockets采用两个唯一的元组识别（源IP/源端口和目的地IP/目的地端口），例如192.168.15.8/3287 + 184.27.235.114/80。这就可以解释为什么运行在端口80上的网络服务器存在多条开放连接。本地（或临时端口）通过由TCP栈分配，而新端口由发起方（客户端）为每个接收方（服务器）连接分配。

有诸多标准端口（您可能认识）可供各类应用使用，包括：

* HTTP 80
* SMTP 25
* DNS 53
* POP 110
* MQTT 1883

这些端口称为“常见端口”并受IETF互联网数字分配机构（IANA）管理。IANA确保为互联网用途设计的任何两个应用不会使用同一个端口（无论是针对UDP还是TCP）。

WICED能够轻松支持TCP Sockets（*wiced\_tcp\_create\_socket()*），同时您可以创建自己的协议供物联网设备和服务器对话，您也可以实施其他人定义的定制协议。

例如构建一套定制协议，我们可以将WICED Wi-Fi示例协议（WWEP, WICED Wi-Fi Example Protocol）定义为ASCII文本协议。客户端和服务器都将发送一个下列形式的字符串：

* 命令Command：代表命令(R=读取,W=写入,A=接受,X=失败)的一个字符。
* 设备ID, Device ID：代表设备十六进制值的4个字符，例如1FAE或002F。每个设备都在服务器上存在自身独有的寄存器集，这样您可以使用唯一的ID（除非您希望将同一个寄存器集当作另一个设备读取/写入）。
* 寄存器, Register：代表寄存器的2个字符(每个设备有256个寄存器），例如0F或1B。
* 值,Value：代表16位无符号整数的六进制值的4个字符。应在“R”命令上忽略该值。

客户端可以发送“R”和“W”命令。服务器采用“A”（并返回数据）或“X”（不返回数据）响应。服务器中包含一个数据库，用于存储写入到服务器的值（在客户端使用“W”命令时）并将回送请求的值（当客户端使用“R”命令时）。服务器随时跟踪每个设备ID的独立256寄存器集。例如ID为0x1234的设备、地址为0x0F的寄存器，与ID为0xABCD的设备、地址为0x0F的寄存器不同。

协议的开放版本运行在端口27708上，安全的TLS版本运行在端口40508上。

部分示例：

* “W0FAC0B1234”将写入值0x1234到ID为0x0FAC的设备的寄存器0x0B。服务器随后使用“A0FAC0B1234”响应。
* “W01234”是一个非法数据包，服务器会以“X”响应。
* “R0FAC0B”读取ID为0x0FAC的设备的寄存器0x0B。在本例中该服务器使用“A0FAC0B1234”响应（第一种情况下写入值1234）。
* “R0BAC0B”是一个合法读取，但因为没有数据写入到该设备，所以服务器会用“X”响应。

注意“raw”Sockets本身不具有安全性。TCP sockets只发送任何通过链路赋予它的数据。在使用安全功能的情况下，由TCP上级层，例如SSL或TLS负责加密/解密数据（具体在后面讨论）。

WICED SDK中提供了Sockets，以便您构建自己的定制协议。但是一般情况下开发人员大都使用第7章所讨论的标准应用协议之一（HTTP、MQTT等）。

### 使用Sockets的WICED-SDK TCP服务器和客户端

在下面的例子中我使用前面章节中定义的WWEP协议来示范使用Sockets在WWEP客户端（198.51.100.14）和WWEP服务器（198.51.100.3）间创建连接的步骤。



上图描述的是在TCP服务器（断续线左侧）和TCP客户端（右侧）两个设备间建立TCP连接所需的步骤。这两个设备已经连接IP网络并且已经分配了IP地址（192.51.100.3和14）。每个系统分为四个部分：

* 您的固件应用（标注有“Application”的方框）。这是您使用WICED-SDK写入，用于控制系统的固件。该固件同时供服务器和客户端使用。
* 负责处理所有通过网络通信的TCP/IP stack。
* 代表65536个TCP端口（编号0-65536）的Port。
* 代表4到1500字节RAM，用于保持“发送”（T）数据包和“接收”（R）数据包的Packet Buffer。

为设置TCP服务器连接，服务器固件应该：

1. 调用下列命令*创建TCPSockets*（Sockets属于一个*wiced\_tcp\_socket\_t类型的结构）：*

*wiced\_tcp\_create\_socket(&socket, WICED\_AP\_INTERFACE)；*

1. 调用下列命令，将Sockets附加到WWEP服务器TCP端口27708：

*wiced\_listen(&socket,27708);//27708是用于WWEP协议的端口号*

1. 调用下列命令休眠当前线程并等待连接：

*wiced\_tcp\_accept(&socket);*

为设置TCP客户端连接，客户端固件将：

1. 调用下列命令创建TCP Sockets：

*wiced\_tcp\_create\_socket(&socket, WICED\_STA\_INTERFACE);*

1. 通过调用下列命令“绑定”到特定TCP端口（可以是任何端口，因此我们指定WICED\_ANY\_PORT，这使TCP/IP栈选择任何可用端口）：

*wiced\_tcp\_bind(&socket, WICED\_ANY\_PORT)；*

1. 要创建到服务器的实际连接，您需要做两件事：
   * + - 1. 找到服务器地址。以类型*wiced\_ip\_address\_t*的WICED数据结构传递服务器地址。我们先假定您已经定义了名为*serverAddress*的该类型结构。

您可以用两种方式之一初始化该结构——静态方式，或使用DNS。

* 要静态初始化它，您可以使用WICED SDK提供的宏，具体如下：

*SET\_IPV4\_ADDRESS(serverAddress, MAKE\_IPV4\_ADDRESS(98,51,100,3));*

* 要通过DNS循环初始化，请进行下列操作：

*wiced\_hostname\_lookup( "wwep.wa101.cypress.com", &serverAddress, 10000 );*

* + - * 1. 现在您有了该服务器的地址，您通过调用*wiced\_tcp\_connect()*并等待TIMEOUT（毫秒）连接到端口27708。在我们的局域网中，TIMEOUT可小于1秒，但在WAN中TIMEOUT可能会延长到几秒钟：

*wiced\_tcp\_connect( &socket, &serverAddress, 27708, TIMEOUT);*

### 使用流发送和接收数据

在连接创建完毕后，您的应用将要在客户端和服务器之间传输数据。最简单的通过TCP传输数据的方式，是使用WICED SDK提供的流功能。使用流功能，您无需了解具体如何把数据打包成统一的数据包，就能发送和接收任意数量的数据（请参阅下一部分对数据包的详细介绍）。

要使用流，您必须首先明确流结构并使用您的网络连接Sockets初始化这个流结构：

*wiced\_tcp\_stream\_t stream;*

*wiced\_tcp\_stream\_init(&stream, &socket);*

这一步完成后，使用*wiced\_tcp\_stream\_write()函数就能方便地写入数据。*该方法将流和消息均视为参数。消息就是一个待发送的字符数组。在您写入到流之后，您需要调用*wiced\_tcp\_stream\_flush()* 方法。下列代码用于写入单条消息：

*char sendMessage[] = “TEST\_MESSAGE”;*

*wiced\_tcp\_stream\_write(&stream, sendMessage, strlen(sendMessage));*

*wiced\_tcp\_stream\_flush(&stream);*

从流读取数据使用*wiced\_tcp\_stream\_read()函数。*该方法将流和消息缓冲视为参数。该函数还要求您指定读入缓存的最大字节数和Timeout。该函数返回一个*wiced\_result\_t值，该值可用于确保读取流的成功。*

*result = wiced\_tcp\_stream\_read(&stream, rbuffer, 11, 500);*

在后台，通过流进行的读取和写入使用一致的数据包。WICED SDK中的流函数会对您隐藏这些数据包的管理，便于您就能集中精力关注您应用的较高级层面。但如果您要对通信施加更强的管控，您可以使用WICED SDK API直接发送和接收数据包。

### (高级)使用WICED SDK以TCP客户端形式用数据包传输数据

在您的应用的开头，当您在控制台上运行*wiced\_init()函数时，您将看到消息“Creating Packet pools”。*Packet pools只是用来存储来自网络的数据包（即接收数据包）或保持外送但尚未送出的数据包（即发送数据包）的RAM buffer。默认条件下有两个接收数据包和两个发送数据包，但是这可以在您的固件中加以配置。如果您用尽了接收数据包，TCP数据包就会被丢弃。如果您用尽了发送数据包，在您创建发送数据包时就会出错。

Buffer中的每个数据包包含：

* 一个分配引用计数
* 原始数据
* 指向数据开头的指针
* 指向数据末尾的指针
* TCP数据包开销

数据包的生命周期最初处于未分配状态，因此引用计数值为0。如果您想要发送消息，您将调用*wiced\_tcp\_packet\_create()*，其原型为：

wiced\_result\_t wiced\_packet\_create\_tcp( wiced\_tcp\_socket\_t\* socket,

uint16\_t content\_length,

wiced\_packet\_t\*\* packet,

uint8\_t\*\* data,

uint16\_t\* available\_space );

该函数将查找未分配的数据包（即引用计数=0）并将它分配给您。命令行参数为：

* *socket*:指向之前由*wiced\_tcp\_connect()创建的Sockets的指针。*
* *content\_length*:表示您计划在数据包中放多少字节数据。
* *packet*: 指向数据包指针的指针。该参数指示创建函数，为您提供一个指向RAM中数据包结构的指针。要使用它，您需要指明：*wiced\_packet\_t \*myPacket；*然后在您调用*wiced\_packet\_create\_tcp()*时，您将传递一个指针给您的指针，例如*&myPacket*。当该函数返回时，*myPacket将指向数据包池中已分配的数据包。*
* *data*：指向uint8\_t指针的指针。和上面一样，这能够让创建函数为您提供一个指向RAM中数据包结构的指针。要使用它，您需要指明：*uint8 \*myData；然后在您调用wiced\_packet\_create\_tcp()时，您将传递一个指针给您的指针，例如&myData*。当该函数返回时，*myData指针将指向数据包缓冲内需要存储您的数据的位置。*
* *available\_space*:这是个指向一个整数的指针，这个整数将设置为允许您在数据包内存储数据的最大数量。它的原理与之前的两个相似，即函数会修改您的整数实例。

当您创建了数据包之后，您需要：

* 将您的数据复制到数据包中正确的位置，即使用*memcpy()复制到之前提供给您的数据位置。*
* 调用*wiced\_packet\_set\_data\_end()*，告知数据包您的数据末尾位置。
* 调用*wiced\_tcp\_send\_packet()发送数据。*该函数会增加引用计数（因此在调用该函数后引用计数会增加到2）。
* 最后您将调用*wiced\_packet\_delete()*，释放对数据包的控制。该函数会减少引用计数。当数据包由TCP/IP栈实际发出，它将再次减少引用计数，使数据包缓冲可供重复使用。

鉴于上述特点，传送固件的内容大致如下：

wiced\_tcp\_socket\_t socket;

wiced\_packet\_t \*myPacket;

uint8\_t \*data;

uint16\_t availableDataSize;

char myMsg[]=”WABCD051234”;

uint16\_t msgLen=strlen( myMsg );

wiced\_tcp\_create\_socket(&socket, WICED\_STA\_INTERFACE);

wiced\_tcp\_bind(&socket, WICED\_ANY\_PORT );

wiced\_tcp\_connect(&socket, &serverAddress, SERVER\_PORT, TIMEOUT);

wiced\_packet\_create\_tcp( &socket, msgLen, &myPacket, &data, &availableDataSize );

memcpy( data, myMsg, msglen );

wiced\_packet\_set\_data\_end( myPacket, data + msgLen );

wiced\_tcp\_send\_packet( &socket, myPacket );

wiced\_packet\_delete( myPacket );

wiced\_socket\_delete(&socket);

在调用wiced\_tcp\_packet\_create\_tcp后：

* 指针*myPacket将指向数据包池中分配给您的数据包。*
* *availableDataSize*将设置为您在数据包中能存储的最大字节数（大约1500）。您应确保向数据包中复制的内容不超过数据包存储量。为了简化本示例操作，我没有在上述代码中做这种检查。
* 指针*data*将指向您需要复制您消息的位置（在这一行中我使用*memcpy完成这一操作）。*

在在您在进行指针运算时，要注意调用wiced\_tcp\_set\_data\_end的那一行。

### (高级)使用WICED SDK以TCP服务器形式接收数据包

作为一个TCP服务器，您可能会有一个进行下列操作的线程：

* 调用*wiced\_tcp\_accept(&socket)函数，让该函数挂起您的线程，等待数据到来。*数据到来时，它将唤醒您的线程并继续执行。RTOS有一个“accept timeout”，默认下在大约3秒后会唤醒您的线程。如果它发生超时，*wiced\_tcp\_accept的返回值将是与WICED\_SUCCESS不同的值。*接着由您选择怎么操作。
* 一旦数据到来，您可以调用*wiced\_tcp\_receive*。该函数的样板为：

wiced\_tcp\_receive( wiced\_tcp\_socket\_t\* socket,

wiced\_packet\_t\*\* packet,

uint32\_t timeout )

*wiced\_packet\_t \*\** packet的意思是指您需要为它提供一个指向类型*wiced\_packet\_t*指针的指针，这样接收函数就能将您的指针设置为指向数据包池中的TCP数据包。该函数还会增加这个数据包的引用计数，因此当您完成操作时您需要调用*wiced\_packet\_delete*删除这个数据包。

* 最后您调用*wiced\_packet\_get\_data*（样版见下）即可得到实际的TCP数据包数据：

wiced\_result\_t wiced\_packet\_get\_data(

wiced\_packet\_t\* packet,

uint16\_t offset,

uint8\_t\*\* data,

uint16\_t\* fragment\_available\_data\_length,

uint16\_t \*total\_available\_data\_length )

该函数在设计上能让您分片捕获数据包，因此具有偏移参数。要取得您的数据，您需要传递一个指针给一个uint8\_t指针。该函数会更新您的指针，将它指向缓存中的原始数据。

综上所述，接收固件的大致内容如下：

while(1)

{

wiced\_packet\_t \*myPacket;

uint8\_t \*myData;

uint16\_t frag\_len,avail\_len;

result = wiced\_tcp\_accept( &socket ); // The thread will suspend until a packet is received

if (result != WICED\_SUCCESS) // Probably a timeout occurred

continue; // Skip the rest of this iteration through the loop

wiced\_tcp\_receive( &socket, &myPacket, WICED\_WAIT\_FOREVER );

wiced\_packet\_get\_data( myPacket, 0, &myData, &frag\_len, &avail\_len );

myData[avail\_len] = 0; // add null termination so we can print it

WPRINT\_APP\_INFO((“Packet=%s\n”, myData));

wiced\_packet\_delete( myPacket );

wiced\_tcp\_disconnect(&socket);

}

该代码段假定您正在接收的是一个短字符串且它能装入一个数据包。而且显然这里不存在错误检查。

注意服务器一旦接收到数据包就会断开与Sockets的连接（它不会删除Sockets，只是断开与它的连接）。这在TCP服务器中属于常见做法，这样在不必要的时候就无需保持Sockets连接。当客户端打开另一个连接时，调用*wiced\_tcp\_accept()能够使服务器接收下一个数据包。*

### WICED Sockets文档

WICED-SDK为您提供了一个函数库，用于开展基于Sockets的通信。关于Sockets的WICED文档位于Components 🡪 IP Communication 🡪 TCP内。其中的API子章节专门介绍了数据包通信、缓冲通信、流通信和服务器通信。我们将主要介绍数据包通信，但高级练习也会覆盖流和服务器API方面。



### (高级)对称和非对称加密：基础知识

由于我们面临着数据包未被加密的问题，那接下来该怎么办？您在的浏览器窗口中会看到“HTTPS”，这里的“S”代表的是安全（Secure）。顾名思义，这是指它对所有通信使用了加密通道。但这又如何实现？您如何才能让安全通道运行？拥有安全通道会的意义何在？安全又是指什么？这是一个非常复杂的话题，因为要形成对加密的基本数学理解，就要擅长高等数学，这不是人人都能做到的。而且本手册篇幅有限难以展开讨论。另外诠释这类知识这也远远超出了我的能力范围。但是不要绝望。让这个功能发挥作用的实际操作其实很简单。

所有的加密实际做的是同一件事情。它先取得未加密数据，将它与密钥结合，运行加密算法取得加密数据。原始数据称为纯文本或明文，加密数据称为“密文”。然后您通过网络传输密文。当另一侧收到数据，它将数据与密钥结合，对密文进行解密，运行解密算法生成明文，即原始数据。

目前存在两种加密方案，分别是对称加密和非对称加密。

对称意即两侧都使用相同的密钥。也就是说您用来加密的密钥和您用来解密的密钥是相同的。这类加密的实例包括AES和DES。因为速度极快且安全，对称加密属于倍受青睐的选择。令人遗憾的是，要使用这种方法，两侧都需要知道这个密钥，因为加密密钥和解密密钥是完全相同的。问题在于，如果两侧从未进行过通信，您如何才能让两侧都知道这个密钥？对称密钥加密的另一个问题就是，一旦密钥丢失或被盗取，整个系统也将受损。

非对称加密方法，往往也称为公共密钥，使用两个数学上相关的密钥。这两个密钥往往称为“公”钥和“私”钥。私钥用于解密由公钥加密的数据，反之亦然。这是一个出色的工具，因为您可以将公钥提供给任何人。他们可以使用您的公钥加密数据，而之后只有您的私钥能解密它。非对称加密的优秀之处在于即便您知道公钥，您也无法推测出私钥（单向函数）。这种加密方法的问题在于它速度慢，需要设备上有较大密钥存储空间（一般用闪存）来存储公钥（例如对PGP而言是192字节）。

现在怎么做？最常见的通信方法是使用公钥加密传递一个私钥对称密钥，这个私钥随后用于其余部分的通信：

* 首先您向服务器开通一个未加密连接
* 将您的公共密钥提供给该服务器
* 该服务器随后创建一个随机对称密钥
* 而后服务器使用您的公共密钥加密它新创建的随机对称密钥并发回给您
* 您使用您的私钥解密该对称密钥
* 您再使用对称密钥加密开通一个新通道



这一方案能够完全有效地对抗窃取行为。但是如果有人窃取原始公共密钥又该怎么办？没有关系，因为他们没有解密对称密钥所需的“客户端私钥”。那缺陷在什么地方呢？这一方案的缺陷在于无法对付所谓的中间人（MIM）。中间人攻击的方式是：

* 您先向服务器[但其实是一个MIM]开通一个未加密连接。
* 您将公钥发送到MIM。
* MIM再向服务器开通一个通道。
* MIM发送自己的公钥到服务器。
* 服务器使用MIM的公钥加密一个对称密钥，将其发回给MIM。
* MIM使用自己的私钥解密对称密钥。
* MIM把用您的公钥加密的对称密钥发送给您。
* 您使用自己的私钥解密MIM对称密钥。
* 然后您使用该对称密钥开通到MIM的新通道。
* MIM使用该对称密钥开通到服务器的通道。

只要MIM居于中间，它就能读取所有流量。只有在MIM在第一次事务中就居于中间，您才会受这种攻击威胁。除此之外的通信就是安全的。

但是如果某人控制了网络中的中间连接点，例如Wi-Fi接入点，MIM就很容易发生。只有两种方法可以防范MIM攻击：

* 预先共享公钥（这样您可以保证您掌握的是正确的密钥）
* 使用认证授权(CA)

CA是互联网上一个拥有大型密钥字典的服务器。要使用CA，您要在自身系统中嵌入由CA认证的公钥（这样您就能与CA建立安全连接）。然后当您从您不认识的某人处获得密钥时，您再打开到CA的安全连接，由CA验证您拥有的密钥是否与发给您密钥匹配。

如果MIM把自己的公钥发送给您，然后您使用CA检查，会发现MIM的公钥不属于您要连接的服务器，然后您就会知道自己正在遭受MIM攻击。在与CA对话时如何防范MIM？通过在您的程序中内建已知有效证书，即可防范MIM。今早我检查我的Mac机里的证书，发现有179个内建的有效证书。

### 安全套接层 (SSL Secure Sockets Layer) / 传输层安全 (TLS Transport Layer Security)

密钥共享要想发挥效果，每个人都必须认可同一种实现密钥交换和获得加密的标准方法。这种方法就是SSL及其后续技术TLS。这是两种应用层协议，用于处理前文提到的密钥交换，在网络服务器或运行MQTT的WICED设备上级的一层提供加密数据管道。SSL是一个相当耗资源（内存和CPU）的协议，已经在相当程度上被较轻量级和更新型的TLS所取代。

这两个协议一般都归属于应用层，但我个人而言我经常感觉它们实际属于应用层和传输层之间。TLS内建在WICED中，如果您在初始化连接时给它提供密钥（来自DCT），它的运行就会对它之上的层透明。下一章中将讨论的几种应用层协议都要依靠TLS连接 - 例如HTTP🡪TLS🡪TCP🡪IP🡪Wi-Fi Datalink 🡪 Wi-Fi 🡪 Router 🡪 Router🡪Server Ethernet🡪Server Datalink🡪Server IP🡪Server TCP🡪TLS🡪HTTP Server。

关于TLS的文档，请参阅Components🡪IP Communication🡪TLS Security。



## 练习

### 01 创建一个IoT客户端，当在客户端上按下按钮时，向运行WWEP的服务器写入数据

根据上文的描述，我们已经使用运行非安全版本WWEP协议的WICED-SDK实现了一个服务器，具体如下：

* DNS名：wwep.wa101.cypress.com
* IP地址：198.51.100.3
* 端口：27708

您的应用将监测板上按钮动作并根据每次按动触发一个LED。此外您的应用还将连接到WWEP服务器并在每次按下按钮时发送LED状态。该应用要求：

* LED的特征数为5。这即是说LED状态存储在256字节寄存器空间中的地址0x05。
* 该LED的“值”为0代表熄灭，1代表点亮。
* 使用您的设备MAC地址的16位校验和作为该设备ID。

应用必须完成的步骤有：

1. 连接Wi-Fi。
   1. 提示：选用之前章节中您的一个项目作为起点。
2. 在uint16\_t格式下相加MAC字节求出您的设备号（实质上的校验和）。
3. 使用DNS获取服务器wwep.wa101.cypress.com的IP地址，或使用INITIALIZER\_IPV4\_ADDRESS和 MAKE\_IPV4\_ADDRESS硬编码IP地址。
4. 将LED初始化为OFF。
5. 设置GPIO以监测按钮。
6. 如果按下了按钮：
   1. 切换LED状态。
   2. 发送数据到服务器。
      1. 格式化您要发送的消息（使用*sprint()*）
         1. ‘W<device number>05<state>’
         2. 提示：<device number>计算见上文。
         3. 提示：<state>对应OFF为‘0000’，对应ON为‘0001’。
      2. 向WWEP服务器开通一个Sockets（创建、捆绑、连接）。
      3. 初始化流
      4. 向流写入您的消息
      5. 刷新流
      6. 删除TCP流（提示：*wiced\_tcp\_stream\_deinit()*)
      7. 删除Sockets
7. 查看WWEP级服务器控制台，确定您的事务已经完成
8. 提示：确保您为任何创建的线程分配律足够大的栈尺寸（6200应足够）

### 02 修改(01)以检查返回代码

注意在WWEP协议中，服务器返回的数据包的第一个字符或为“A”，或为“X”。该练习中，请阅读来自服务器的响应并确保您的原始写入操作正确。分别使用一个合法数据包和一个非法数据包测试。

提示：可通过调用“*wiced\_tcp\_stream\_read()*”完成

### (高级) 03修改(02)以使用TCP数据包

在本次练习中您将重复练习02以创建自己的数据包，而不必依赖流函数。遵循练习02的各个步骤设置本练习，初始化您的网络连接并创建一个Sockets。然后传输数据到服务器时，不创建流，改为采用下列步骤：

1. 创建一个有11个字节的传输数据包：
   1. ‘W’ (写入命令)
   2. 采用4字节的六进制编码ASCII字符作为您的设备ID
   3. ‘05’– 两个ASCII字符代表LED特征的寄存器编号
   4. ‘0000’或‘0001’–4个ASCII字符代表“OFF”和“ON”
   5. 提示：使用*sprint()函数格式化消息*
2. 向Sockets发送数据包
3. 删除传输数据包
4. 从服务器读回数据
   1. 提示：使用wiced\_tcp\_receive()和 wiced\_packet\_get\_data()
5. 将接收到的数据打印到终端。
6. 删除接收到的数据包。
7. 删除Sockets。
8. 查看WWEP级服务器控制台，确定您的事务已经完成

### (高级) 04 在服务器侧实现非安全WWEP协议，每次处理一个连接(使用TCP数据包读取)

提示：参阅API指南中关于TCP服务器通信的部分。

提示：对数据库使用链接好的清单，这样数据库能在无条目的情况下启动，然后随着数据存入增加容量。

提示：WICED库提供了一个链接清单工具程序，可在libraries/utilities目录下找到。您可以直接使用#include“linked\_list.h”将它纳入内容，这个命令也会提供API文档。

### (高级) 05 在服务器侧实现非安全WWEP协议，每次处理一个连接(使用TCP回调)

提示：查看函数wiced\_tcp\_register\_callbacks。您将需要回调函数进行Sockets连接、从客户端接收到的新数据和Sockets断开连接。

提示：您可能需要将传送和接收缓存池大小提高到6，这样同时发生的连接将不会造成服务器上的缓存溢出。这些功能的设置文件请查看“43xxx\_Wi-Fi\ \platforms\BCM94343W\_AVN\BCM94343W\_AVN.mk”。

需要注意的行有：

GLOBAL\_DEFINES += TX\_PACKET\_POOL\_SIZE=2 \

RX\_PACKET\_POOL\_SIZE=2

注意使用本平台的工作空间中每一个项目都是用该文件，因此它会影响到您的全部项目。就我们的示例而言，我们不会使用大量SRAM，但要注意的是每个数据包都约为1500字节，所以从SRAM占用的角度而言，增大数据包池是成本高昂的。

### (高级) 06 修改 (05) 以一次性处理多个连接

### (高级) 07 修改 (03) 在40508上使用TLS安全功能

### (高级) 08 在服务器侧实现安全的WWEP协议

## 补充阅读

[1] RFC1700 – “Assigned Numbers”；互联网工程任务组 (IETF) - https://www.ietf.org/rfc/rfc1700.txt

[3] IANA Service Name and Port Registry - http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml