Lab 6 - Capture the flag

CC3501

Task Description

You have purchased an internet-connected embedded system, but the product comes with rather limited software. You have decided to develop your own software to interact with the vendor's official web service. However, the vendor does not document the communication protocol. You emailed them asking for technical documentation and were rudely rebuffed. The vendor even stated that you wouldn't be able to create your own client because there is a secret key used to identify "genuine" clients. All in all, this sounds like a challenge.

You have decided to reverse-engineer the protocol and find the secret key, so that you can write your own client. Your initial investigations have revealed the following facts:

1. The software is distributed as a Linux binary compiled for ARMv7, which is the CPU architecture of the Raspberry Pi.
2. The secret key is constructed algorithmically at run time, so it cannot be found simply by running the strings command on the binary.
3. The software uses HTTP to communicate with a network server.

Your task is to crack the secret key and write your own client to transmit messages to the server.

The provided app allows people to write messages to a virtual message board, but the message is always "Official app says hello". **Your task is to develop a program that enables any other message to be posted.** You will achieve this by intercepting the network traffic generated by the official app to reverse engineer the protocol (including the secret key).

Steps to follow

1. Download the "lab6-official-app" binary from LearnJCU and copy it to your Raspberry Pi. Note that to run it, you probably will need to give execute permissions:

   ```
   $ chmod +x lab6-official-app
   ```

   Experiment with it to understand how to use it, e.g. to read messages from the server as well as write messages to the server. Note that the application is a command line tool that explains its usage when you run it from the terminal, e.g.

```
$ ./lab6-official-app
```

Once you understand the operation of the official client, you are ready to proceed with intercepting its communication.

2. Install the wireshark application, which will be used for intercepting the network traffic generated by the application.

```
$ sudo apt install wireshark
```

You can answer either way with the question about whether non-root users should be allowed to capture packets. The easiest is to just run wireshark as root.

3. On a graphical desktop (e.g. VNC), run wireshark as root:

```
$ sudo wireshark
```

4. Capture network traffic. If you are using VNC then there will be a lot of network traffic due to VNC, so you don't want that noise cluttering up your capture. Therefore, you can filter to select only HTTP traffic by using the capture options setting in wireshark, as shown in Figure 1. Make sure that you choose the interface that your Pi is using to access the Internet, i.e. etho for Ethernet or wlano for Wifi.

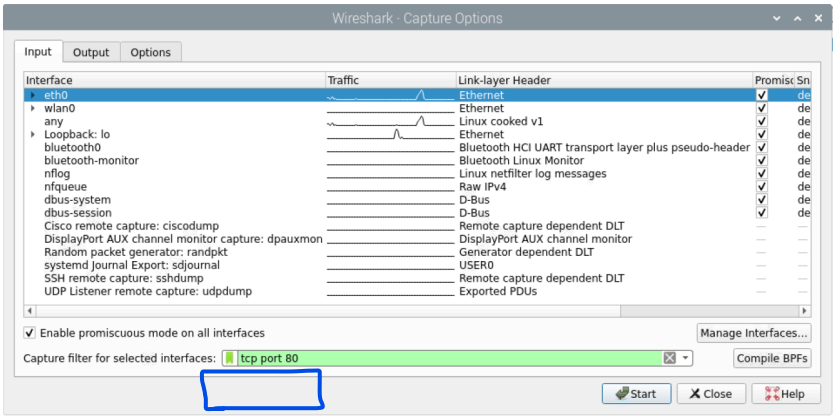


Figure 1: Capture options with the filter "tcp port 80" so that only HTTP traffic will be captured.

5. While Wireshark is recording traffic, use the official app to post a message to the server. Confirm that you can see the HTTP transaction in Wireshark.

6. Examine the network traffic. Wireshark will decode the HTTP transaction to display the "Full request URI".

>Inspect the HTTP Request:

>Click on the packet that contains the GET request.

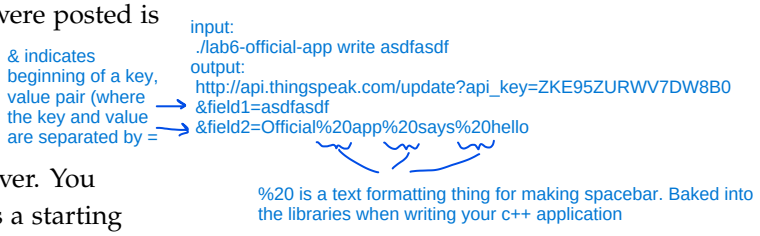
> (Uniform Resource Identifier). This is the complete URL that was requested, including the domain, path, and any query parameters.

7. Test the URI in a web browser to confirm a proof-of-concept that you are able to post messages to the server.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	192.168.0.204	3.92.67.51	TCP	74	53796 → 80 [SYN] Seq=0 Win=32120 Len=0 MSS=1460 SACK_PERM TSval=3225739634 TSecr=0 WS=1…
2	0.001280014	3.92.67.51	192.168.0.204	TCP	74	80 → 53796 [SYN, ACK] Seq=0 Ack=1 Win=14480 Len=0 MSS=1460 SACK_PERM TSval=1818682700 T…
3	0.001332902	192.168.0.204	3.92.67.51	TCP	66	53796 → 80 [ACK] Seq=1 Ack=1 Win=32128 Len=0 TSval=3225739635 TSecr=1818682700
4	0.001430585	192.168.0.204	3.92.67.51	HTTP	207	GET /update?api_key=ZKE95ZURWV7DW8B0&field1=asdfasdf&field2=Official%20app%20says%20hel…
5	0.002637175	3.92.67.51	192.168.0.204	TCP	66	80 → 53796 [ACK] Seq=1 Ack=142 Win=15616 Len=0 TSval=1818682700 TSecr=3225739635
6	0.582287006	3.92.67.51	192.168.0.204	HTTP	604	HTTP/1.1 200 OK (text/plain)
7	0.582313061	192.168.0.204	3.92.67.51	TCP	66	53796 → 80 [ACK] Seq=142 Ack=539 Win=31872 Len=0 TSval=3225740216 TSecr=1818682758
8	0.582461799	192.168.0.204	3.92.67.51	TCP	66	53796 → 80 [FIN, ACK] Seq=142 Ack=539 Win=31872 Len=0 TSval=3225740216 TSecr=1818682758
9	0.583914198	3.92.67.51	192.168.0.204	TCP	66	80 → 53796 [FIN, ACK] Seq=539 Ack=143 Win=15616 Len=0 TSval=1818682758 TSecr=3225740216
10	0.583987049	192.168.0.204	3.92.67.51	TCP	66	53796 → 80 [ACK] Seq=143 Ack=540 Win=31872 Len=0 TSval=3225740218 TSecr=1818682758

Hint: the easiest way to confirm that the messages were posted is to use the official application to read them back, i.e.

```
$ ./lab6-official-app read
```



8. Write a C++ application to post messages to the server. You can use the HTTP demo from last week's lectures as a starting point. Your application must accept the username and message as command-line arguments.

   - Hint: If your message contains spaces, wrap it in quotes on the command line so that it will be passed to your application as one argument.
   - URL parameters cannot contain special characters (they must be represented by a percent sign then the ASCII code, e.g. %20 for a space). The libcurl function linked below can perform this conversion for you:

     `https://curl.se/libcurl/c/curl_easy_escape.html`
   - You can concatenate strings nicely in C++ using the std::string type and the + operator.

Assessment

To complete this lab, produce your own custom client app that successfully posts messages to the server, where the username and message are delivered on the command line.

You do not need to replicate the read functionality of the official app, only the write mode.