In order to be able to successfully set-up a Bluetooth sniffing system you will need the following:

1. An Ubertooth One device.
2. An additional Bluetooth dongle or Bluetooth adapter.
3. A PC which runs Windows, Mac, OSX or Linux.
4. A bootable USB running Kali Linux (this one is optional)

You can perform these activities on just about every major desktop operating system but for this tutorial we focussed on running this with Kali Linux.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

**The most difficult part of Bluetooth sniffing – installing all of the tools**

I caution you – there are a number of resources on the internet dedicated to setting up the Ubertooth One, this post included. After spending a day and a half attempting to get bluetooth sniffing working with absolutely zero background on the subject, my perseverance and can do attitude kicked in and I had the Ubertooth and associated scanning commands working like a charm. In order to save everybody the trouble here are the golden rules of getting the Ubertooth running on your machine:

1. Ignore all of the blog posts and websites which tell you how to install / configure the Ubertooth One (most of them are no longer relevant).
2. Only follow the installation guide at [this location](https://github.com/greatscottgadgets/ubertooth/wiki/Build-Guide" \t "_blank)
3. Update your Ubertooth firmware asap.

One final tip make sure you have a range of additional Bluetooth tools you can use for Bluetooth debugging / sniffing if you’re having trouble with the install its good to have an arsenal of other tools you can use to verify if the issue you are experiencing is a configuration issue or a hardware issue.

**Learning to walk before you can run, viewing the Wi-Fi spectrum with the Ubertooth On**

Getting the Ubertooth One setup for Bluetooth sniffing isn’t the easiest activity to get up and running so I recommend before you go into any sniffing you walk through some of the Ubertooth Ones other capabilities i.e. spectrum analysis.

To get started make sure you have installed Kismet and spectools for spectrum analysis. To install kismet and spectools follow the commands below:

root@kali:~# git clone https://www.kismetwireless.net/spectools.git Cloning into 'spectools'... root@kali:~# cd spectools root@kali:~/spectools# ./configure root@kali:~/spectools# make root@kali:~/spectools# make install

Now to run Spectools simply type:

root@kali:~/spectools# /usr/local/bin/spectool\_gtk

Below are some screen caps of some of the Spectrum analysis I performed:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

**How does one listen in on Bluetooth?**

Once your Ubertooth is setup and configured you can run the following commands to analyse Bluetooth traffic.

hcitool is one of the default linux Bluetooth utilities when running it will return the MAC address and name of any Bluetooth devices in range:

root@kali:~# hcitool scan

One of the Ubertooth utilities is ubertooth-scan also allows you to passively monitor Bluetooth traffic – below is the command line usage:

root@kali:~# ubertooth-scan –s

The majority of the scanning work I undertook was with using ubertooth-btle. This allows you to capture Bluetooth traffic between Bluetooth low energy compatible devices this is becoming one of the preferred methods of Bluetooth communication between new devices, next time you’re in a store just look at the Bluetooth devices most of them should say Bluetooth-low energy compatible.

To run ubertooth-btle in promiscuous mode and output the contents int a pcap file simply run the command:

root@kali:~# ubertooth-btle –p –f -c capture.pcap

Below is a sample of what the btle packets look like when your run ubertooth-btle in promiscuous mode:

systime=1441512979 freq=2440 addr=8d651b4d delta\_t=3.599 ms 86 9e d1 00 65 92 86 01 5d 3e 0e 5e 65 e0 61 9a 7d f7 04 c8 9f f5 45 00 ce f5 cc c8 8f 67 02 f5 4f a7 f5 Data / AA 8d651b4d (valid) / 30 bytes Channel Index: 17 LLID: 2 / LL Data PDU / L2CAP start NESN: 1 SN: 0 MD: 0 Data: d1 00 65 92 86 01 5d 3e 0e 5e 65 e0 61 9a 7d f7 04 c8 9f f5 45 00 ce f5 cc c8 8f 67 02 f5 CRC: 4f a7 f5 systime=1441512979 freq=2440 addr=72f844df delta\_t=146.421 ms 01 00 9b 72 68 Data / AA 72f844df (valid) / 0 bytes Channel Index: 17 LLID: 1 / LL Data PDU / empty or L2CAP continuation NESN: 0 SN: 0 MD: 0 Data: CRC: 9b 72 68

Ubertooth-btle in action

If you want to follow a specific device you can use the command below where “00000000” would be the devices MAC address:

root@kali:~# ubertooth-btle –a 00000000

**Using crackle to decrypt Bluetooth packets**

Crackle is an easy to use brute force cracking utility. Since most Bluetooth pairing codes which encrypt traffic between two devices are only 4-6 digits long they can be easily decrypted if the pairing between two devices is captured and this also allows you to listen in on future interactions between the two devices as we will show shortly, but fist lets walk through installing crackle which can be done by following the commands below:

root@kali:~# tar xf crackle-0.1.tgz root@kali:~# cd crackle-0.1 root@kali:~/crackle-0.1# ls aes.c aes-enc.c aes\_i.h COPYING crackle.h README aes-ccm.c aes.h AUTHORS crackle.c Makefile test.c root@kali:~/crackle-0.1# make cc -Wall -Werror -g -c -o crackle.o crackle.c cc -Wall -Werror -g -c -o aes.o aes.c cc -Wall -Werror -g -c -o aes-ccm.o aes-ccm.c cc -Wall -Werror -g -c -o aes-enc.o aes-enc.c cc -Wall -Werror -g -c -o test.o test.c cc -o crackle crackle.o aes.o aes-ccm.o aes-enc.o test.o -lpcap root@kali:~/crackle-0.1# make install

Once crackle is installed we can begin walking through using crackle to decrypt pcap files with Bluetooth data on them. To do this simply run the following command on your desired pcap file:

root@kali:~/crackle-sample# crackle -i ltk\_exchange.pcap -o decrypted.pcap TK found: 000000 ding ding ding, using a TK of 0! Just Cracks(tm) Warning: packet is too short to be encrypted (1), skipping LTK found: 7f62c053f104a5bbe68b1d896a2ed49c Done, processed 712 total packets, decrypted 3

Now to listen in on future communications between the two devices run the following command on a pcap file and supply the LTK value you discovered earlier:

root@kali:~/crackle-sample# crackle -i encrypted\_known\_ltk.pcap -o decrypted2.pcap -l 7f62c053f104a5bbe68b1d896a2ed49c Warning: packet is too short to be encrypted (1), skipping Warning: packet is too short to be encrypted (2), skipping Warning: could not decrypt packet! Copying as is.. Warning: could not decrypt packet! Copying as is.. Warning: could not decrypt packet! Copying as is.. Warning: invalid packet (length to long), skipping Done, processed 297 total packets, decrypted 7

**Issues**

With some of my own BLE captures there wasn’t enough packet info to run crackle successfully so I ran crackle with some sample files to give an overview of how the tool works.

**Resources**

Here are a few good resources you should check out:

* http://j2abro.blogspot.com.au/2014/06/understanding-bluetooth-advertising.html
* http://j2abro.blogspot.com.au/2014/06/analyzing-bluetooth-advertising-with.html
* http://cerescontrols.com/tutorials-3/sniffing-bluetooth-packets-with-kismet-and-wireshark-in-ubuntu-12-04/
* https://github.com/greatscottgadgets/ubertooth/wiki/Build-Guide
* https://github.com/greatscottgadgets/ubertooth/wiki/Capturing-BLE-in-Wireshark
* http://stackoverflow.com/questions/23877761/sniffing-logging-your-own-android-bluetooth-traffic
* https://lacklustre.net/bluetooth/wireshark.html
* https://blog.lacklustre.net/posts/BLE\_Fun\_With\_Ubertooth:\_Sniffing\_Bluetooth\_Smart\_and\_Cracking\_Its\_Crypto/
* http://superuser.com/questions/947593/how-can-i-sniff-bluetooth-traffic-coming-from-my-and-another-device
* http://www.backtrack-linux.org/forums/showthread.php?t=41552
* http://www.splitbits.com/2014/05/14/ubertooth-spectools-chromebook/
* http://ubertooth.sourceforge.net/usage/start/
* http://hackerific.net/2012/01/28/Spectrum-Tools-and-Ubertooth-One/
* https://penturalabs.wordpress.com/2014/02/20/ubertooth-updated-for-2014/
* https://blog.lacklustre.net/

**Conclusions**

Bluetooth, hopefully this article packages it up into a nice and easy digestible format as the process to get it to this level wasn’t that straight forward or error free.

Bottom of Form