Busting The Bluetooth® Myth – Getting RAW Access

aka “Transforming a consumer Bluetooth® Dongle into a Bluetooth® Sniffer”

Max Moser
http://www.remote-exploit.org

Introduction

During the last year, rumours had come to my attention that apparently it is possible to transform a standard 30USD Bluetooth® dongle into a full-blown Bluetooth® sniffer. Thinking you absolutely need Hardware to be able to hop 79 channels 1600 times a second I was rather suspicious about these claims.

This paper is the result of my research into this area, answering the question whether it is possible or not.

Analyzing Drivers

I used 4 different dongles during my tests, and these used the very same chipset from CSR. However I noted that the features they offer were different and as such assumed that it must be the firmware that offers most of them.

For an overview about what is actually required to promiscuously sniff Bluetooth® I downloaded commercial software that is freely available to everyone and inspected the files that come with the packages. Within the INI[1] files I stumbled across drivers for a chip made by CSR (Cambridge Silicon Radio). In a specific section there are all the devices listed including their unique USB® vendor ID (VID) and product identifier (PID).

A regular CSR BlueCore[2] device has the value:

"USB\VID_0A12&PID_0001"

By further analyzing the files available in the commercial Bluetooth® sniffer package, I recognized that the driver used within that package identifies itself as:

"USB\VID_0A12&PID_0002"

The difference being only the digit at the end of the VID. I now have the VID the commercial sniffing tool seems to be expecting.

Analyzing Other Content

Within the installation directory of the unnamed commercial Sniffer package, I spotted .dfu[3] files which appeared to be some sort of firmware files.

Finding Useful Target Dongles

After finding references to CSR driver/chipsets in the installation package I goggled for CSR based Bluetooth® dongles.
It is not that easy to find one which is for sure CSR based but eventually I found a few and purchased them.

Hint : When you insert a Bluetooth® dongle into your linux box, you can use "lsusb" or "usbview" to show all connected usb devices. I was supprised that 2 of my 4 dongles are showing me a familiar value of:

0xa12:0x0001 Cambridge Silicon Radio

Analyzing CSR Chipset And Its Abilities

By searching through the CSR website for more information I discovered a lot about the Implementation of the various Bluetooth® features in their chipsets, and I recognized that the chip has different “stores” (Memory).

I suddenly remembered a Bluez tool called btaddr which can change a Bluetooth® USB dongle BTaddress, so I wondered whether the ProductID can be changed using the same or similar techniques.

Soon I realised that by using the tool bccmd from the bluez CVS tree, I can completely read and partially write to the dongles different storage areas, including the areas where the Bluetooth® vendor and product id are stored!

I gave it a try and successfully modified my PSF store to hold now the desired values of:

0xa12:0x0002

1 http://en.wikipedia.org/wiki/INI_file
2 http://www.csr.com/products/bcrange.htm
3 http://acronyms.thefreedictionary.com/Device+Firmware+Upgrade

Installing The Drivers In Windows

Using my modified dongle I tried to install the drivers supplied by the commercial software's viewer version. And it did work! The drivers recognized the dongle as a genuine part of the sniffer product package.

Flashing The Dongle With The Commercial Firmware

Wondering whether there is a way to upload the dfu files I found in the installation package onto the dongle, I came across software called dfutool, also part of the Bluez utilities. I tried to flash the commercial firmware onto the stick and guess what.... no errors... I was shocked. It seemed like the stick is now flashed with their firmware version.

I re-inserted the stick in my Linux computer and did see RAW as feature within my hciconfig output, in addition I see the RX and TX number of bytes rising.

So now we have an exact copy of the commercial hardware sniffer, with the correct firmware, correct vendor and product ID. One question remains “Will it sniff?”.

Luckily I was able to find a person that owns a licensed version of the sniffer and finally was able to test it.

I found out that prior to using the dongle I have to configure it with their configuration tool. This was not as easy as planed, but after changing the MAC address of my modified dongle to the same value as the licensed one, it was working as expected.

Conclusion

- Most stuff is not done in hardware but software, that was a widely spread myth
- The price is not a hurdle for Black hats
- It should be possible to code a Linux sniffer

Resume

I like to state here very clear, that I did this all for educational research purposes and am quite shocked that this all was possible. It seems that the rumours are true and sniffing Bluetooth® is not a matter of expensive hardware, but of proprietary firmware and software.

This means that the Bluetooth® is much more vulnerable to sniffing than expected for months and that this security through obscurity approach might have opened the gates for the Black hats discovering holes before we do.

About the Author

Max Moser is the founder of remote-exploit.org and works currently for Dreamlab Technologies Ltd. as Security Analyst and Tester.