Breaking the Security of Physical Devices

Dr Silvio Cesare

Qualys

1 Introduction

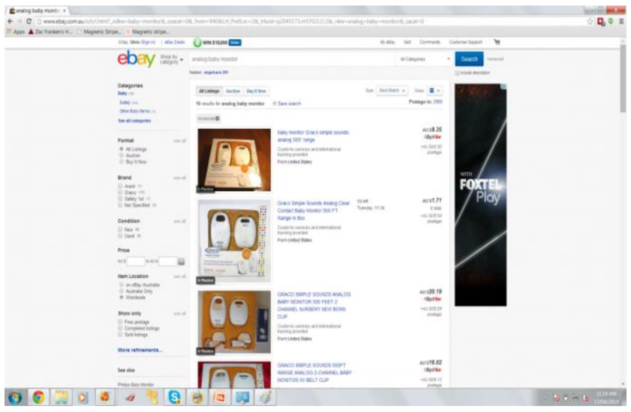
Electronics, computers, mobile phones, microcontrollers are ubiquitous devices in the world we live in. These devices, previously disparate things, are now converging. This allows the security researcher who previously only knew about IT security, or the researcher who only knew electronics, to apply their knowledge across a number of devices which share all these atttributes.

This whitepaper looks at some of those physical devices that we commonly see and applies a variety of hardware, software, and radio techniques to break their security. The devices in question examined here include:

1. Baby monitors
2. Home alarm systems
3. Automobiles

It is hoped that this paper inspires researchers to look at other real-world devices.

2 Eavesdropping on analog Baby Monitors



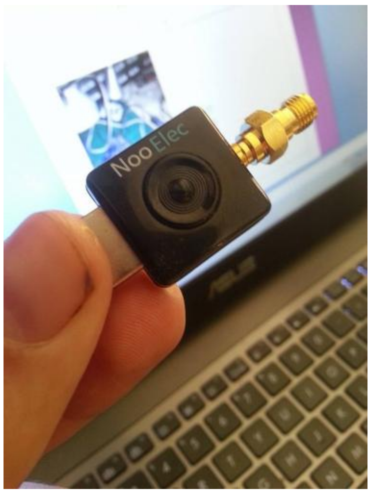
Baby monitors are common. There are a variety of types that exist, such as IP-based baby monitors, or DECT-based baby monitors. A pre-cursor to IP and DECT-based systems is the analog baby monitor which transmits and receives a radio audio signal. Analog baby monitors are easily obtainable and can be quickly found and purchased on E-Bay.

A baby monitoring system works by putting in a listening device in a room with the baby. At a specific frequency an audio signal will be continuously transmitted to another room in where a receiver takes the RF signal and emits the audio.

Software defined radio (SDR) represents an ideal solution to tackle eavesdropping on an analog baby monitor.

Many types of software define radio devices and dongles exist. The Funcube Dongle [1] is one such device. A cheaper device is based around the TV tuner range of SDRs. These SDR dongles cost little more than $15 [2] and can receive but not transmit on a wide range of frequencies.

Also required is an antenna and these can be cheaply purchased on E-Bay. The frequency range of SDR devices can be expanded to receive lower frequencies by using an upconverter. An upconverter moves the signals in lower frequency ranges into a frequency range that the SDR dongles can process. One such upconverter is the Ham It Up [3] upconverter.



Determining what frequency the baby monitor is transmitting at is done by using a spectrum analyser. Expensive professional devices are available in conjunction with cheaper hobbyist devices. One such inexpensive device is the RF Explorer [4]. What I found was that the analog baby monitor in my possession was transmitting at 40MHz.

Now that the transmitting frequency is known, we need to demodulate the signal back to audio. Software spectrum analysers such as HDSDR [5] on Windows or Gqrx [6] on Linux are useful. These programs can both show signal strength and demodulate the signal using AM or FM.

All the pieces of the puzzle are complete, and I know that the signal transmitted by the baby monitor is at 40MHz and through testing the different modulation schemes I found that FM modulation was used. The software spectrum analysers produce audio from the baby monitor and it can be heard quite clearly.

It is certainly possible to transmit on the baby monitor frequency, but using the cheap SDR dongles we can only receive.

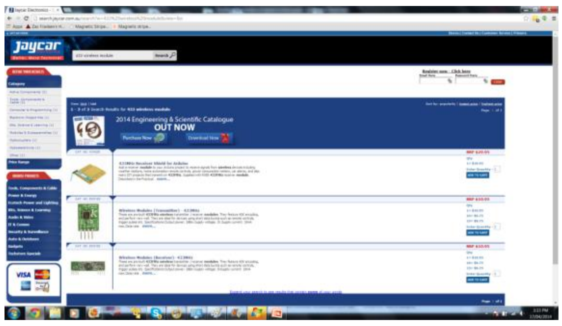
Mitigating eavesdropping attacks can be done by upgrading the baby monitors to IP or DECT-based solutions. DECT has known problems also, however implementations of those attacks are not wide-spread.

3 Disabling and tampering with RF-based home alarm systems

Many home alarms available on the market are controlled by a remote control that enables and disables the security systems. Such home alarms can be bought from home depot and home hardware stores and most have remote control functionality. These remote controls use RF and can be analysed using software defined radio.

The security on home alarm systems is not high and often used fixed RF codes to enable and disable the system. Fixed codes are susceptible to replay attacks. An attacker listens to the RF signal waiting for the user to disable the system. The attacker captures this signal, and then later when performing the live attack, replays the captured code. The alarm system is then disabled.

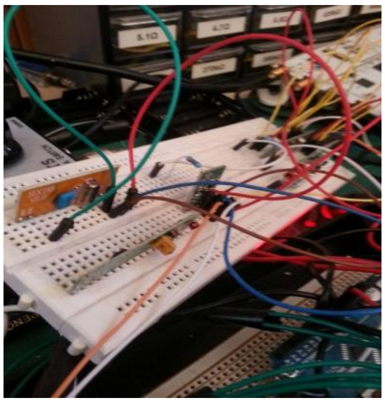
This attack works on most home alarm systems. The standard frequencies that the keyfobs use are 315MHz and 433.92Mhz.



The har dware required to do this attack is different to the baby monitoring attack. An SDR device that can transmit as well as receive is required. A common SDR device is the USRP [7] and the B200 model is used here.

GNURadio [8] is the software component of the attack that is used. In capturing the alarm code, the USRP is used as a signal source and a File is used as a sink. To replay the codes, the source is the File and the sink is the USRP.

What exactly is the signal that is generated and how do we know that the signal is using a fixed code? To answer this we need to be able to demodulate the signal. Generally, most keyfobs use a combination of AM and PWM to modulate the signal. The presence of an AM carrier wave for a specific length of time designates whether the data being sent is a binary 0 or a binary 1. Typically, a long pulse represents one value, and a short pulse represents another.



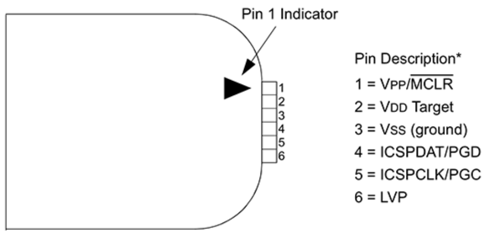
The pulses used in pulse width modulation (PWM) can be measured and grouped together into clusters. This forms the basis of knowing what the length of a pulse is. Cluster analysis is used as implemented in the pycluster [9] Python library. The mean of each cluster is the length of time for that pulse cluster. The mean of the two means (for 0,1) designates a threshold to classify a pulse as being short or long.

A USRP costs close to $1000. A cheaper solution to implement this attack is to buy commercial AM transmitter and receiver modules. These modules cost a hundredth of the price of a USRP. The modules give or take data that can be provided by an Arduino development board. For $50 it is possible to build a device as I have done using these components to implement replay attacks against home alarms.

To mitigate replay attacks, it is advisable to buy a commercial-grade alarm system. Typically these systems will not use fixed codes, but codes that change each time they are sent. This is an example of a rolling code. Avoid home depot and hardware shops and pay for a professional installation if security is important.

The next attack looks at tampering. One home alarm system that was being tested was dismantled in a physical attack. I discovered that the main IC on the board was clearly labelled as a PIC microcontroller. Test ports were nearby, and an immediate possibility was that the test ports were for device programming. I soldered header pins onto the board and did a continuity test was on each pin to discover ground.

PICKIT™ 3 PROGRAMMER CONNECTOR PINOUT



* The 6-pin header (0.100" spacing) accepts 0.025" square pins.

The ground pin that was found matched the ICSP (in circuit serial programming) interface used to program PIC microcontrollers. At this point, I attached a device programmer was attached ran device programmer software.

What I found was that the device programmer identified the microcontroller in the alarm system. The software was not able to read the firmware on the device as a fuse had been set to protect the code. Howe ver, the data could be read and this data revealed the secret password used to program the keypad on the home alarm system.

Ideally, it would be nice to read the firmware from the micocontroller. Some possible attacks include

silicon analysis after decapsulation. Another attack could include glitching the device while attempting to read the firmware.

It is hard to stop attackers with physical access. The marking on the IC could be removed that would make it more difficult for an attacker. It should be assumed that physical tampering and reversing may be attempted but it is very hard to stop a well resourced attacker.

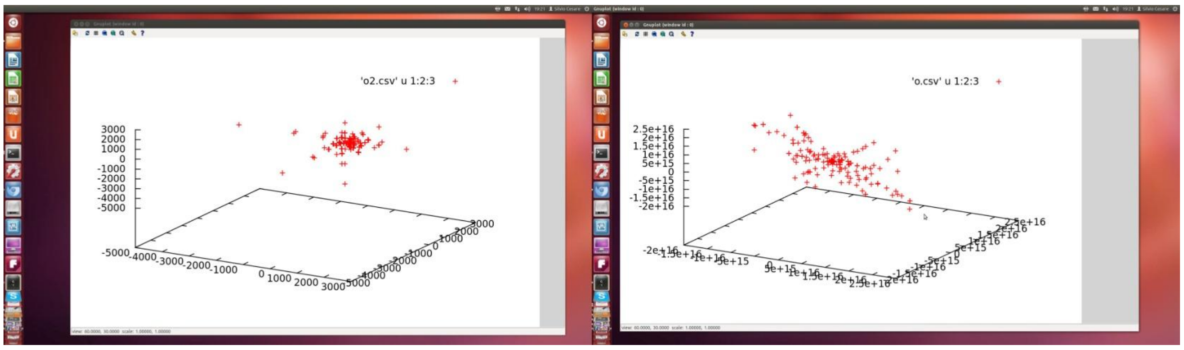
4 Defeating the keyless entry of a 2000-2005 model automobile

Automotive remote keyless entry systems represent attractive targets to analyse. The keyfobs can be analysed using software defined radio and the security of these devices has been largely investigated. Perhaps the best known attack was the cryptographic break of the Keeloq [10] rolling code which was used in a large number of remote control systems, including (supposedly) automobiles.

I investigated the security of a popular 2000-2005 model vehicle. The reason to examine this particular automobile was that one was in my possession. This model car is documented as still being manufactured in particular countries. From my front driveway, I can see other cars of the same make and model on my street.

The first method of analysis was to generate a dataset of what happens when a button on the remote is pressed. At first, I simply tried pressing the button repeatedly for a period of time. This was effective, but repetitive and time consuming. I decided to automate this. To do this, I built a button pushing robot. This was effectively a solenoid controlled by an Arduino development board. The button presses were simultaneously analysed using SDR equipment - a B200 USRP and GNURadio.

The keyfob transmits at 315MHz which is typical of a keyfob. It is noted that holding down a button on the remote only transmits a single code. On other types of remotes, holding down a button on



the remote repeats the code that is being transmitted for the duration of the button press. The codes transmitted use AM and PWM modulation schemes.

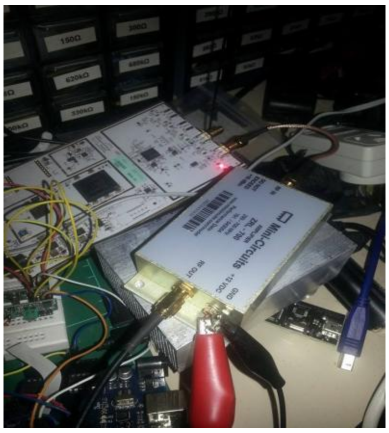
The code transmitted by the remote is not fixed and changes each time it is sent. A digit in any part of the code can be one of three states, a 1, a 0, or a gap. A gap is important and is part of the code. A 0, and a gap are the same size in terms of timing, and a 1 is twice that size. An implicit gap also follows every 1 or 0.

It is clear that there exists a preamble which is the same for every transmission. This preamble includes synchronisation data and then a code which tied to the individual remote as an identification. The preamble also contains a section for defining which button the remote was pressed, either lock, unlock, trunk, or panic. Finally, the remainder of the code is the non-fixed part of the code that changes each time a button is pressed.

Because the code changes each time, we can assume this particular remote uses a rolling code. The way a rolling code works is that the remote, which only transmits, and the car, which only receives, both maintain a pseudo random number generator (PRNG). The state of the PRNG on both the keyfob and the car are synchronized. Therefore, the car can predict the next pseudo random number that the keyfob creates. When a button on the remote is pressed, it transmits this pseudo random number and the car can verify it is correct. Receivers typically allow for the next X valid numbers to authenticate the device. This is done so that if the keyfob is pressed less than X times and the car does not receive the transmission, the car will still accept the remote as authentic.

The PRNG of the keyfob can be analysed using the dataset of button pushes created earlier. I used phase space analysis [11] to do this analysis. Phase space analysis was used famously for analysing the initial sequence numbers of TCP connections and the software to do this analysis is freely available. This software was used in the analysis against our automobile. The idea is to transform a sequence of numbers into a sequence of 3-dimensional points in space. These points are created by taking the difference between numbers in our original dataset as the value along a particular axis. If the PRNG is strong, the 3-dimensional visualisation should produce a uniform cloud of points. If the PRNG is not strong, the cloud will appear to be non random.

Phase space analysis reveals that the sequence of numbers in the car's PRNG can be predicted with a high degree of success given the previous 3 numbers. Therefore, to perform an attack against the car, we to eavesdrop and capture the last 3 codes, then predict the next rolling code using phase-space analysis and transmit the code using the USRP and GNURadio.



To increase the range of the transmitter, I use an amplifier in the output stage of the system that is powered by a bench supply. We also need to test that the codes transmitted by the USRP can be accepted by the car before implementing the attack with predicted codes. The way I validated this was by performing a capture and replay of the real remote. If the car is in range of the remote, capturing the code will make it invalid once it has been received by the automobile. I prevented the car from receiving the code by putting the remote inside a Faraday cage. A Faraday cage is an RF shield that blocks RF. Faraday cages can be made as metal cages, or metal screened cages. An inexpensive version can be simply a $1 Aluminium foil lined freezer bag. I used 2 bags, one inside the other, and this worked effectively. Using the capture and then transmitting the code I want, I verified that the transmission component of the attack potentially works.

The actual attack is effective and the predicted rolling codes are correct.

A better attack would be a bruteforce of the rolling code. If we look at the rolling code, we can see that the total timing of the entire transmission is one of two sizes and the transmissions alternate between them. There are also fewer gaps than 1s or 0s. We also note that a gap never follows a gap.

A theoretical bruteforce attack would capture any 1 transmission by the real remote to obtain the correct preamble. At that point, all numbers in the rolling code range would be generated using the constraints above. This gives us less than 1 million guesses.

I implemented the bruteforce and the car successfully unlocks typically under 2 hours.

There is something funny happening though when I implement this attack. Some of the codes always unlock the car. What this appears to be, is a manufacturer backdoor that bypasses the rolling code security. It now takes seconds to unlock the car once the backdoor is known.

The backdoor is tied into the identification of the remote. This identification is in the preamble of the transmissions, but at this point, I have been unable to determine the algorithm to generate the backdoor given a transmission capture. However, brute forcing can find the backdoor.

It's hard to mitigate attacks against cars without a recall. It is possible to replace the keyless entry with a more secure system.

Needless to say, newer cars have stronger security, so upgrading the car is als o an option. For car makers, a PRNG could be replaced with a list of codes stored in flash memory that is generated by a true random number generator. Also, obviously, avoid coding in backdoors.

In the future I would like to continue this research by doing a silicon level analysis after decapsulating the IC in the keyfob. This could potentially help in recovering the firmware and finding the algorithm to generate backdoors.

Conclusion

Hardware hacking is fun. Many real-world devices can be attacked using hardware and radio techniques. This leaves much room for analysis by security researchers. Rolling codes and keyfob security is a prime area ready for more investigation and it seems likely that many systems could be vulnerable.

References

[1] *Funcube Dongle*. Available: http://www.funcubedongle.com/

[2] *SDR*. Available: http://www.nooelec.com/store/sdr/sdr-receivers/nooelec-nesdr-nano-sdr-dvb-t-usb-stick-r820t-w-antenna-and-remote-control.html#.U7YG5vm1ZZU

[3] *Ham It Up*

Available: http://www.nooelec.com/store/ham-it-up-v1-0-rf-upconverter-for-software-defined-radio.html#.U7YGufm1ZZU

[4] *RF Explorer*. Available: http://rfexplorer.com

[5] *HDSDR*. Available: http://www.hdsdr.de

[6] *Gqrx*. Available: http://gqrx.dk

[7] *USRP*. Available: http://home.ettus.com

[8] E. Blossom, "GNU radio: tools for exploring the radio frequency spectrum," *Linux journal,* vol. 2004, p. 4, 2004.

[9] *pycluster*. Available: http://bonsai.hgc.jp/~mdehoon/software/cluster/software.htm#pycluster

[10] N. Courtois, G. Bard, and D. Wagner, "Algebraic and Slide Attacks on KeeLoq," in *Fast Software Encryption*. vol. 5086, K. Nyberg, Ed., ed: Springer Berlin Heidelberg, 2008, pp. 97-115.

[11] M. Zalewski. *Strange Attractors and TCP/IP Sequence Number Analysis*. Available: http://lcamtuf.coredump.cx/oldtcp/tcpseq.html