

HACKING WIRELESSLY-CONTROLLED GARAGES USING SOFTWARE DEFINED RADIO

Hailin Yu, Won Ho Chung

yhltc@ucla.edu, wonhochung@ucla.edu

ABSTRACT

This project presents how to exploit wirelessly-controlled garages/gates using software defined radio (SDR), RTL-SDR, and HackRF One device. Furthermore, it investigates in-depth how these features work, and how they can be spoofed. As wide variety of electronics devices used these days utilize radio frequency (RF) to receive and transmit signals for the communication, this work can be applied to many different applications, such as listening to radio communication, hacking the car key, and even hijacking the drones. This work will focus on how to conduct key-space reduction attacks on fixed-codes and extend its implementation technique to show how to protect from these kinds of attacks.

Index Terms— Radio Frequency, Modulation, GQRX, HackRF One, Wirelessly Controlled Garages, RTL-SDR, Recording Signal, Receiving (RX) signal, Transmitting (TX) Signal.

1. INTRODUCTION

As more electronics devices are being connected, greater attention in how to properly transmit, store, and protect the signals has also increased rapidly. For example, wirelessly-controlled remote control keys provide convenience to users when getting in and out of one's property by eliminating the physical action of turning the key or rolling-up the garage door manually. These kind of devices utilize radio frequency (RF) signals as the medium for communication. Even though most of people tend to be careless about how to secure the communication of these signals, getting these signals hacked are equivalent to getting their home keys or sensitive personal information being stolen. Therefore, this project of hacking wirelessly-controlled garages using software defined radio aims to present the implementation how vulnerable these RF communication can be and hope to increase the public's awareness regarding consequent issue.

The whole project was on github:
<https://github.com/Alspeakeryhl/hacking-wirelessly-controlled-gates-using-hackrf>

2. RELATED WORK

Almost every electronic devices that uses radio frequency to transmit and receive the signal can be exploited. With the proper devices with properly configured software, people with minimal signal processing knowledge could perform or re-implement our method presented. Among many possible related-attacking applications using software defined radio, one of the well-known incident using RTL-SDR was tracking planes. It is possible to track commercial aircraft flights and output their locations to mapping software to see exact real-time coordinates. Furthermore, this relatively inexpensive SDR device allows users to tune into an enormous range of frequencies, including FM radio, fire and police bands, aircraft communication, and even digital TV.

While RTL-SDR is only capable of receiving the signal, HackRF can both receive and transmit the signal in which could perform even more different applications. For example, HackRF can be used to perform replay attack against certain vehicle models that are already in the market. The replay attack is a very trivial attack that can easily be performed just by recording the signal, and then rebroadcasting (replayed) again. Normally, wireless car locks use rolling code that transmits different codes every time user presses as the security measures to prevent such an attack. According to the related work found [11], the attacker verified one of the car model from one domestic car company was prone to this replay attack where the attacker simply recorded the unlock and lock signals using HackRF with GNU radio.

3. ATTACK METHOD

There are mainly two kinds of attacks for fixed code which is easier than rolling code. We will introduce both in this paper, while for implementation we only try first one. The first method is to record the signal when someone is pressing the key to open a gate, then transmit recorded signals to open the gate. The second one is called Brute Force method, in which you just need to know the frequency of the code, then you can try all combinations of the key, for example, for a 10-bit code, you need try 1024 different combinations. This will not take plenty of time, if you have a good way to create different combinations, actually for 8-12 bit key, you only need 8 seconds to try all the possible situations.

3.1. FCC ID





First of all, we need to know the frequency of our remote key. FCC ID number contains a grantee code (three in length) and an equipment product code (five in length), which is assigned to identify all devices. When you go to fcc.io and input the FCC ID of the device, then you can find the information of it including key's frequency. The information of our device is shown in Fig.1 and Fig.2, we can find that our key has three different possible frequency: 300MHz, 310MHz and 390MHz.



Fig. 1. FCC ID Information of the Garage Remote Key

3 results were found that match the search criteria:
Grantee Code: EF4 Product Code: NE02X4

Displaying records 1 through 3 of 3.

View Form	Display Exhibits	Display Grant	Display Correspondence	Applicant Name	Address	City	State	Country	Zip Code	FCC ID	Application Purpose	Final Action Date	Lower Frequency In MHz	Upper Frequency In MHz
				Nortek Security & Control LLC	9919 Sea Otter Place	Carlsbad CA	United States	92010	EF4NE02X4	Change in Identification	09/10/1999	300.0	300.0	
				Nortek Security & Control LLC	9919 Sea Otter Place	Carlsbad CA	United States	92010	EF4NE02X4	Change in Identification	09/10/1999	310.0	310.0	
				Nortek Security & Control LLC	9919 Sea Otter Place	Carlsbad CA	United States	92010	EF4NE02X4	Change in Identification	09/10/1999	390.0	390.0	

Perform Search Again

Please use the Submit Inquiry link at www.fcc.gov/labels to send any comments or suggestions for this site

Phone: 888-CALL-FCC (325-5322)
TTY: 888-TELL-FCC (835-5322)
Fax: 202-418-0232
E-mail: fccinfo@fcc.gov

Fig. 2. Device Information After Looking Up It's FCC ID

3.2. Recording and Transmitting

When we know the frequency of the device, then we can do recording and transmitting near the target frequency.

First, we use Gqrx to see if we can receive signals. Fig.3 and Fig.4 shows the recording key in Gqrx with different devices. From them, we can see that the target frequency is around 297MHz. We can see that for RTL-SDR device, it will not only record real signals, but also some noise besides it, while for HackRF, almost no noise will be recorded.

There are two recording methods in Gqrx, one is to record signals as wav files, the other is to record I/Q raw data as

images. However, since the recording sample rate is really small (48kHz), the wav files didn't work for our purpose.

Next, we try to record and transmit signals by using macports and Hackrf. The set up codes are shown in Appendix. By using them, we can choose the target frequency, large bandwidth (10MHz) and large sample rate (20MHz). Furthermore, it can use RX/TX gain so we can receive and transmit signals farther. The RX gain is up to 62dB and TX gain is up to 47dB which are large enough.

After recording the data, we try to transmit it. To do it, we use HackRF device for transmitting recorded signals and use RTL-SDR to see whether we can receive it. When we add gain to received signals, we can receive signals transmitted from HackRF as strong as the one from remote key.

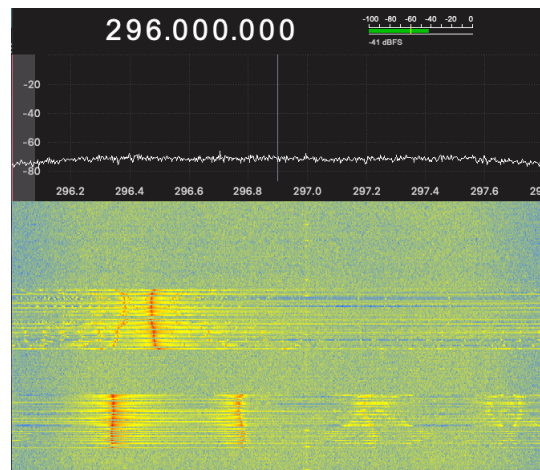


Fig. 3. Recording by RTL-SDR

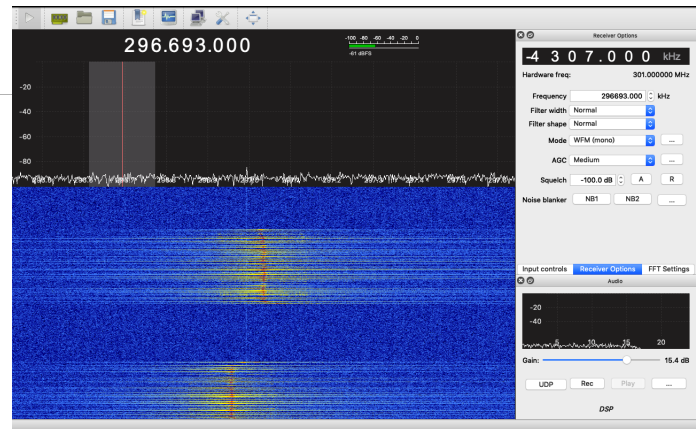


Fig. 4. Recording by HackRF One

3.3. Brute Force method

The first method is easy to implement, however, it needs a long time and plenty of storage for recording signals. The

main drawback is that you need to wait near the gates until someone is coming and pressing the key. Usually, it needs 10-15 minutes in day time and it will create a 10GB big recording file if you are recording all the time. The file is too big and it is a little bit long for you to wait outside a gate, it's very strange and will cause others' attention.

Therefore, we need a better method which only need several seconds and a small storage. This can be done by using so called Brute Force method.

First, we need to decode the signals, it is done in Audacity and shown in Fig.5 and Fig.6. In Fig.5, the above one is the code of gate 1 and the other one is the code of gate 2. Obviously, the above code is 0101010101 and the other one is 1010101010. Also, we expand them to see the detail waveform, and we find that they are just simple sinusoidal function and the only difference for 0 and 1 is the length. As a result, we can just record 0 and 1 each once and then try different combinations of them.(for a 10-bit code it has 1024 different combinations)

Therefore, the total time for a 10-bit code is $1024 * 10 * (2\text{ms signal} + 2\text{ms delay}) * 5 \text{ transmissions} = 3.3\text{minutes}$. It is a short time now. Can we do better? Actually we don't need 2ms delay and the key also works which will reduce time by half. Furthermore, we don't need to repeat five times like the key does. So the time is further reduced by five times. Last but not least, we can use a bit shift register since we don't need to repeat same parts every time. For example, for 2-bit code, we only need to transmit five bits: 00110 instead of eight bits: 00011011. As a result, for 8-12 bits code, we only need $(2^{12} + 1) * 2\text{ms} = 8\text{s}$ to try all possible combinations.

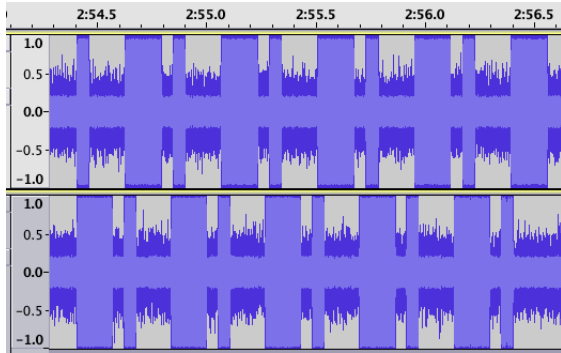


Fig. 5. Decoding signals, there are signals and intervals in it. The length of signal plus interval every bit is same. Shorter signal refers 0 and longer one refers 1. The above one is the code for gate 1 and the other is the code for gate 2

4. EXPERIMENTS AND RESULTS

This part shows all the tools we have used in the experiment and all the results we got during the implementation, all detail codes are provided in Appendix. A video demo can be found

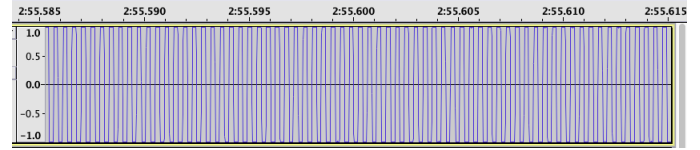


Fig. 6. Expanding signals, this figure shows the detail waveform of signals in Fig.5, since the waveform of code 0 and code 1 are exactly same, we just show one of them

on Youtube:

<https://www.youtube.com/watch?v=AyWHjEQe2uIlist=PLS-uN80-WMVOZ0rezjDUyfE8fMq3DGgDc>

4.1. Tools

For hardware, we have used RTL-SDR(Fig.7) and HackRF(Fig.8). The first one is cheap(about 20\$), it can only receive signal with frequency between 24 and 1766 MHz. Also, it can record I/Q raw data. The second one is expensive(about 300\$), it can both record and transmit signals with frequency ranging from 1MHz to 6GHz. Both devices have open source software.



Fig. 7. RTL-SDR device

For software, we have used Gqrx to figure out whether we can receive the signal and re-transmit it. It can only receive signals and then implement FFT plot so we can see the frequency of the signal. Furthermore, it can save recorded signals to wav files and I/Q raw data. However, the sample rate of it is really small(48kHz),so it will loss plenty of infor-



Fig. 11. Gate1, this photo shows our first gate inside the garage, it is a vertical opening structure with an exposed antenna

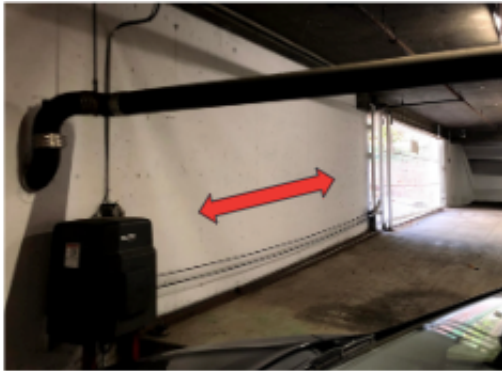


Fig. 12. Gate2, this photo shows our second gate inside the garage, it is a horizontal opening structure and we can't find where the antenna is

signals is 62dB and that of transmitting signals is 47dB. So now the command for recording becomes:
 hack-transfer -r test.bin -f 297000000 -b 2000000 -g 62
 the command for transmitting becomes:
 hack-transfer -t test.bin -f 297000000 -b 2000000 -x 47
 With them, we can now receive signals as far as the exposed antenna in gate 1, and we can transmit codes farther than the remote key. Fig.14 shows terminal information when recording signals, you can see size of data recording every second, the transmitting information is similar. As a result, now we can open gate 2 without knowing where the antenna of it is. Fig.15 and Fig.16 show the long distance we can achieve now.

4.5. Results

Finally, this work was able to open the two garages doors at more than five meters away which is more than the maximum distance that original remote key can achieve. Also, we have verified our attacking implementation outside of the garage door to present the work in more real-life settings.



Fig. 13. The distance between device and exposed antenna when opening the gate1

```
hailindeMacBook-Air:~ yhlts$ hackrf_transfer -r right.bin -f 297000000 -b 100000
00 -g 62
call hackrf_set_sample_rate(10000000 Hz/10.000 MHz)
call hackrf_baseband_filter_bandwidth_set(10000000 Hz/10.000 MHz)
call hackrf_set_freq(297000000 Hz/297.000 MHz)
Stop with Ctrl-C
19.9 MiB / 1.001 sec = 19.9 MiB/second
19.9 MiB / 1.000 sec = 19.9 MiB/second
20.2 MiB / 1.004 sec = 20.1 MiB/second
19.9 MiB / 1.004 sec = 19.8 MiB/second
^Ccaught signal 2
5.2 MiB / 0.254 sec = 20.6 MiB/second
```

Fig. 14. The terminal information when recording signals with gain. For a 10MHz bandwidth it need record 20MB data every second



Fig. 15. The working distance after using gain for gate1

5. CONCLUSION

In this work, we successfully demonstrated the hacking implementation of wirelessly-controlled garages with software-defined radio (SDR). We have verified its performance that it



Fig. 16. The working distance after using gain for gate2

is capable of receiving and transmitting the signals very far away, even farther than the original remote key. Furthermore, for a brute force method, it only takes several seconds to open the gate without waiting nearby until people with remote key access enters. Another method that could increase the security could be requiring the preamble word for beginning of each key, this is very similar to increasing the key space, and it is easier to implement. As a result, the attackers should add preamble word for every combination which will cost a longer time but not highly recommended.

To prevent such attacks, there are several methods that enforces the security of signal transmission. Among many, rolling code algorithm are widely used and recommended these days in cases like car keys. Since rolling code transmits different signals every time user presses the key, it is much more difficult to hack these kind of security method. In order to successfully spoof the signal, it would need to jam the first signal and let the user to press once more to save the second signal for future use.

Therefore, to prevent from getting wirelessly-controlled keys being hacked, we strongly recommend using the devices with rolling codes. For a rolling code, it sends out a different code every time you use it. So even if the attackers are capable of recording it, it would be much harder for them to decode the signal to open the gates. In this project, by presenting the vulnerability of these kind of devices, we hope that public's awareness regarding the security of wirelessly-controlled key has been awakened and increased.

6. REFERENCES

- [1] Radio Hacking: Cars, Hardware, and more! - Samy Kamkar - AppSec California 2016
<https://www.youtube.com/watch?v=1RipwqJG50c>
- [2] Digital Ding Dong Ditch Prank - hacking wireless doorbells w/Arduino and RTL-SDR

<https://www.youtube.com/watch?v=BnwBdeQB7vQ>
<http://samy.pl/dingdong/>

- [3] OpenSesame - hacking garages in seconds using a Mattel toy
<https://www.youtube.com/watch?v=iSSRaIU9-Vc>
<http://samy.pl/opensesame/>
- [4] How to Hack a Garage Door in Under 10 Seconds and What You Can Do About It
<https://www.itstactical.com/intellicom/physical-security/how-to-hack-a-garage-door-in-under-10-seconds-and-what-you-can-do-about-it/>
- [5] Software Defined Radio with HackRF, Lesson 11: Replay
<https://www.youtube.com/watch?v=CyYteFiIozMfeature=youtu.be>
- [6] SDR/HackRF One: Max Setup and Basics
<https://www.jeffreythompson.org/blog/2015/10/11/sdrhackrf-one-mac-setup-and-basics/>
- [7] I finally got my HackRF and I have a Mac Now what?
<https://k1fm.us/2014/08/i-finally-got-my-hackrf-and-i-have-a-mac-now-what/>
- [8] GNU Radio Installation
<https://wiki.gnuradio.org/index.php/MacInstall>
- [9] GNU Radio Companion Installation (GUI tool for signal flow graph)
<https://wiki.gnuradio.org/index.php/GNURadioCompanion>
- [10] Step-by-step: How to install GNU Radio on your Mac OS X with RTL-SDR support
<https://aaronischer.com/wireless-com-SDR/MacOSX-install-gnu-radio.html>
- [11] Using A HackRF To Perform A Replay Attack Against A Jeep Patriot
<https://www.rtl-sdr.com/using-a-hackrf-to-perform-a-replay-attack-against-a-jeep-patriot/>

7. APPENDIX

In this part, all the necessary codes for implementation are documented.

7.1. Macports installation

- 1.install Xcode and the Xcode Command Line Tools
- 2.Agree to Xcode license in Terminal:
`sudo xcodebuild - license`
- 3.Install MACPorts for your version
for macOS Mojave v10.14:
<https://distfiles.macports.org/MacPorts/MacPorts-2.5.4-10.14-Mojave.pkg>

4. Open up a Terminal and run the following commands in it:
sudo port install gr-osmosdr +full
sudo port install gr-fosphor
The step 4 may take a long time.

sudo port install Gqrx
To install gnuradio:
sudo port install gnuradio

7.2. HackRF usage

First, connect HackRF One to your computer, and input `hackrf-info` in the command line to see whether it succeeds. If you see the following information, you can use it now.
Found HackRF board.

Board ID Number: 2 (HackRF One)

Firmware Version: git-44df9d1

Part ID Number: 0x00594f49 0x00594f49

Serial Number: 0x00000000 0x00000000 0x4xx 0x2xx

Next, you need input the following instruction to record your signal as 'test.bin', remember it will be saved in your main path. The optional parameters are all shown.

`hackrf-transfer -r test.bin`

optional parameters:

-i if-freq-hz Intermediate Frequency (IF) in Hz [2150MHz to 2750MHz].

-o lo-freq-hz Front-end Local Oscillator (LO) frequency in Hz [84MHz to 5400MHz].

-m image-reject Image rejection filter selection, 0=bypass, 1=low pass, 2=high pass.

-a amp-enable RX/TX RF amplifier 1=Enable, 0=Disable.

-p antenna-enable Antenna port power, 1=Enable, 0=Disable.

-l gain-db RX LNA (IF) gain, 0-40dB, 8dB steps

-g gain-db RX VGA (baseband) gain, 0-62dB, 2dB steps

-x gain-db TX VGA (IF) gain, 0-47dB, 1dB steps

-s sample-rate-hz Sample rate in Hz (8/10/12.5/16/20MHz, default 10MHz).

-n num-samples Number of samples to transfer (default is unlimited).

-c amplitude CW signal source mode, amplitude 0-127 (DC value to DAC)

-b baseband-filter-bw-hz Set baseband filter bandwidth in MHz. Possible values: 1.75/2.5/3.5/5/5.5/6/7/8/9/10/12/14/15/20/24/28MHz

Finally, you can transmit your recording signal, there are two lights on one sides of the device, when they are on, it means the device is transmitting the signal. The command is shown below, which has the same optional parameters of recording command.

`hackrf-transfer -t test.bin`

7.3. Gqrx and Gnuradio installation

If you want to see the waveform of signal received by hardware or save it as wav files, you can use Gqrx. Furthermore, if you want to decode the signal and see the waveform of transmitted signal, you can use gnuradio. The installation is very easy by using the following command:

To install Gqrx: