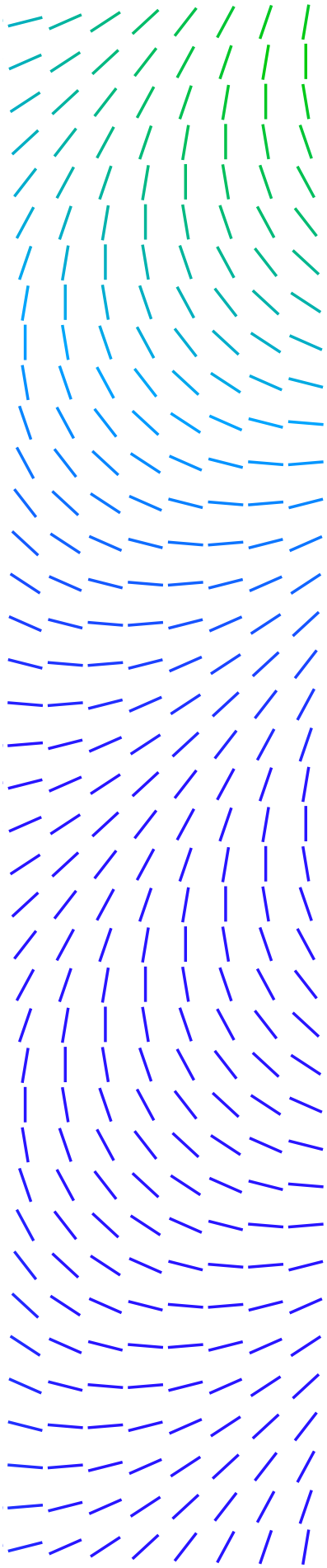

Trellix

FHSS Selective Jamming and Capture of Chambelain Smart Garage Hub "MyQ" State Sensor (MYQ-G0301-E/ MYQ-G0302)

Authors

D. Kevin McGrath Advanced Threat Research
McAfee LLC
Hillsboro, USA
D.Kevin_McGrath@McAfee.com

Sam Quinn
Advanced Threat Research
McAfee LLC
Hillsboro, USA
Sam_Quinn@McAfee.com

Steve Povolny
Advanced Threat Research
McAfee LLC
Hillsboro, USA
Steve_Povolny@McAfee.com

Trellix

Table of Contents

Abstract

Garage door openers have historically used a single carrier frequency with simple off-keying modulation. After being shown vulnerable to a relatively simple jam and replay attack, manufacturers have responded with attempted mitigations, including frequency hopping and multiple modulations. In this work we show that these mitigations are insufficient to the task. Using a full duplex software defined radio, we can selectively jam all used frequencies simultaneously, while capturing the valid signals with the same radio. We also demonstrate a stealth technique for JIT (just in time) style jamming – jamming signals are only broadcast as necessary, without betraying our presence in the absence of garage door activity.

Keywords—RF, SDR, jamming, IoT, "replay attack"

Introduction

Jamming attacks have existed for years, across myriad targets in numerous industries including telecommunications such as radio or tv broadcasting, military such as radar for missile tracking, automotive for key fobs, and many more. While the techniques and applications have evolved over the decades, the underlying concept has not; the goal is to disrupt radio communications by selectively jamming or transmitting in the same frequency range as the target.

As a vulnerability research organization, McAfee Advanced Threat Research (ATR) performs research and analysis on targets spanning multiple industries. One such industry is the consumer, who is increasingly looking for new technology to improve, automate, or replace mundane tasks. A perfect example of this is package courier partnerships with Chamberlain, one of the most popular garage door opener manufacturers The Chamberlain MyQ garage door opener provides delivery personnel a device capable of accessing registered homeowner's garages to deliver packages check happens before any network transactions take place. securely. This capability is delivered via commands to the MyQ API which subsequently communicates directly with the MyQ hub inside of the garage over the local WiFi.

One well-known industry technique for RF jamming is selective jamming, a technique shown to work by Spencer Whyt [1], later dubbed "RollJam" by researcher Samy Kamkar [2], who built a simple, dedicated device allowing for simultaneous blocking of transmission of a rolling code while capturing it for later replay. Our method utilizes the selective jamming concept, while expanding on it to account for the fact that the MyQ state sensor transmits over multiple frequencies simultaneously using a technique known as frequency hopping spread spectrum, or FHSS.

Instead of jamming or capturing rolling codes, we apply it to the "state" of the garage door (ie open or closed). To our knowledge, the techniques described here are the first example of selective jamming and signal capture over multiple simultaneous frequencies.

The goal of the research is to identify means in which cyber criminals could defeat this scenario to gain access to a protected area such as a garage. The demo we built reflects the most common real-world example; a courier using the service to open the garage door, while the attacker device captures the state of the garage (open/closed) as it simultaneously jams the true state of the door sent back to the user, who is away from home. The result is that the user is presented with incorrect information on the state of the door, on their mobile device MyQ app. Because the user can use the app to open or close the garage remotely, they will unwittingly complete the rest of the attack, sending an open command that appears to be closing the garage, and giving full access to the garage to the attacker.

Background

A. Radio Frequency Introduction

In the words of a PhD electrical engineer who focuses on electromagnetic waves "RF is really at the spooky end of the spectrum." In this section we will attempt to lay out the necessary knowledge for understanding this work.

Radio frequency (RF) encompasses the portion of the electromagnetic (EM) spectrum suitable for communication. Lying roughly in the range of 3KHz to 300GHz, well below the visible spectrum (430-750 THz), different portions of the band are known by different terms, from extremely low frequency (ELF) at the low end to extremely high frequency (EHF) at the high end. The MyQ state sensor lies at the lower end of the UHF range, centered on 312.5MHz.

This work makes use of multiple concepts from the field of RF engineering. In order to capture only the portion of the spectrum we care about, we make use of both low pass filters and band reject filters. We also need to ensure we avoid aliasing. All three concepts are briefly explained below.

1. Low pass filters

Low pass filters are one of a class of frequency attenuation filters. In a low pass filter, signals with frequencies lower than the cut off frequency are passed through the filter unaltered, with higher frequencies severely attenuated.

One of the important characteristics of a low pass filter as realized in GNU Radio is the symmetric aspect of it, within both the real and complex domains. That is, a low pass filter can be used to realize a symmetric filter around our central frequency such that frequencies more than a specific distance away are attenuated below our noise flow. In other words, we could use a low pass filter to put an outer limit on the frequencies which riseabove the noise floor.

For additional information regarding RF filters, a reasonably introductory level source can be found in [3] .

2. Band reject filters

Band reject filters are related to low pass filters and exist in the same class of attenuating filters. However, being a reject filter, they attenuate all frequencies which exist between the upper and lower cutoffs.

We make use of band reject filters within this work to attenuate the portion of the spectrum between our frequencies of interest. By combining the low pass filter above with a band reject filter, we can place outer limits and remove unwanted interior noise within the spectrum about which we care.

3. Aliasing

Aliasing, as we are using the term, is the phenomenon of a single signal reappearing in unexpected locations. The specific physical phenomenon is caused by insufficient bandwidth in our output to contain the data within our signal. This shows up in a panadapter as double the number of signals expected, with no way to determine which are real and which are reflections.

This is an artifact of the digital sampling necessary within a software defined radio and its interaction with the continuous nature of the RF wave and the discrete nature of the digital domain of a computer, discussed below.

A real-world example with some of the mathematics background information can be found in [4] .

B. Software Defined Radio

Software defined radios (SDRs) have exploded in popularity in recent years, likely due to the availability of inexpensive hardware. Broadly categorizable into two groups (receive only, transmit capable), these SDRs have inspired a whole new generation of hobbyists within the realm of RF.

There are several important concepts to consider when selecting a software defined radio. These concepts are discussed below.

1. I/Q samples

The traditional view of a signal is instantaneous: at any moment in time, a signal has a magnitude and a phase angle. Traditionally represented in polar coordinates, this instantaneous view doesn't work well for non-immediate use. Storing data in this fashion renders certain types of calculations impossible – questions such as "what's the power of the signal?" and "what's the frequency of the signal?" simply can't be directly inferred with certainty (the latter due to signal mixing, which won't be covered here).

At their core, SDRs are direct I/Q samplers. I/Q data is a mapping of the amplitude and phase angle of the signal to time varying Cartesian space. In other words, we plot the signal in 2D space for every point in time, then stack these planes in time order. This results in a fully 3D representation of the signal, allowing full information to be stored for later analysis. Determining power and frequency are straightforward, and we can reconstruct the full signal at any point in time up to the resolution at which we sampled it.

For additional resources and a more formal description of I/Q data, please see [5] and [6] .

When selecting an SDR, one must consider the sample rates necessary to obtain the bandwidth desired. Higher quality SDRs can provide more I/Q samples per second, and therefore view more of the RF spectrum at once.

2. Sample rate and bandwidth relationship

Given that we are sampling in the digital domain, we are unable to do so at infinite precision. Analog signals exist in a continuous, infinitely precise fashion, which when converted to the digital domain must be sampled in a discrete fashion. How accurately we can reconstruct the signal in question is dependent upon how often we can sample it.

The relationship of bandwidth being sampled and the rate at which we must sample it is known as the Nyquist-Shannon sampling theorem, which states that accurately reconstructing an N Hz channel requires sampling at 2N samples per second. In simple terms, that means we should sample at twice our bandwidth in order to accurately reconstruct the signal.

In practice, the USRP used in this work uses a driver which equates bandwidth and sample rate for implementation simplicity for the user. Internally, the USRP samples at 640 MSps (mega samples per second) and performs extensive signal processing on-chip in order to provide a usable signal to the user such that sample-rate is equal to RF bandwidth.

3. Full- vs half- duplex

The last major consideration from a purely technical perspective is whether the radio can broadcast simultaneously with reception. A full-duplex radio has the capability of transmitting while receiving, while a half-duplex radio does not.

Full duplex radios typically will make use of multiple antennae. A single RF antenna simply can't receive and transmit at the same time. As such, we have to choose between a full duplex radio with multiple antennae (which is typically more expensive than a half-duplex radio) or multiple half-duplex radios, each with their own antenna. In this work, we pursue the former option for its simplicity and overall lower cost.

C. GNU Radio framework

The "software" term in software defined radio isn't just there as technobabble. In order to make use of an SDR, you will need some sort of software running on a host PC – or embedded CPU, external microcontroller, etc. There is an external digital control device for the radio, rather than just analog control of the transceiver. One of the most commonly used frameworks is GNU Radio.

GNU Radio is an open source signal processing framework which provides for both simulation tools and a block-based approach to controlling software defined radios. Providing multiple language-based interfaces, the native libraries are in python. Additionally, a block-based visual programming environment is provided in the form of GNU Radio-companion (GRC).

Several important aspects are worth calling out explicitly regarding GRC and GNU Radio. Specifically, the flowgraph style interface used by GRC, signal probes, and the ability to set CPU affinity of any given block.

1. Flowgraph interface

GRC provides a flowgraph based approach to programming and running an SDR – see Figure IV-6 for an example. Based on the concepts of sources and sinks, data is transformed as it passes through blocks in a direct flow fashion. One of the difficulties of this approach is that performing additional analysis for which there isn't currently a block requires creating a new block for use in GRC.

This flowgraph style interface provides a much more straightforward starting point than having to write raw python source code. By handling the set up and parameterization of the radio and the signals, it can often serve as a useful tool to create the "boilerplate" python code which is used in every program accessing a given radio.

2. Signal probes

One of the most important blocks from the perspective of the current work is the use of a signal probe. These blocks provide an asynchronous way to monitor the signal strength at any point in the flowgraph.

This ability is used to great effect regarding our JIT style jamming. By monitoring the signal level, we can construct our system in such a way as to only broadcast a jamming signal when there is another signal we would like to jam.

Signal probes have an update frequency in GRC which defaults to 100Hz. Tuning this to provide the best experience is often an engineering exercise.

3. CPU Affinity

Software based signal processing is computationally expensive. While modern operating systems can help alleviate this expense via advanced process scheduling strategies, manually dictating which CPUs are used can sometimes result in higher performance of your signal processing algorithms. This is especially true on hyperthreaded systems – preventing common blocks from sharing hyperthreaded logical cores from a single physical core can lead to significant performance improvements.

CPU affinity provides this capability. Unlike most Linux command line tools for setting CPU affinity, rather than a bitmask it takes a core number. Every block provides this capability and taking advantage of the known topology of your CPU can result in enhanced performance.

Threat Model

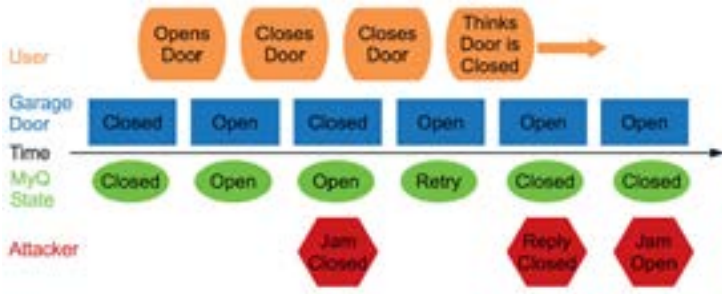

Figure 1: Attack timeline

An attacker may use the technique described in this paper to gain access to a victim garage as follows. Because the MyQ hub has no way of determining the current state of the garage door, the hub is paired with a remote door sensor. This remote door sensor will determine the current state of the garage door based on the physical orientation of the sensor. The remote door sensor is attached directly to the garage door so when the sensor is vertical the door is closed and when the sensor is horizontal the door is open. This remote sensor is set to beacon the current state of the garage door in two cases. The first case is when the sensor has detected that the orientation has been modified, and the second case is to validate that everything is still in sync. To make sure everything is still in sync the remote door sensor will beacon the current state every 30 minutes to confirm the state stored in the MyQ hub.

These beacons are very important for this attack on the MyQ garage door. Since we can jam and capture signals an attacker could start a classic RollJam attack by capturing the latest signal and sending the previous one.

The second important factor in the attack against the MyQ is that someone would have to activate the garage door either locally, remotely, or via third party (such as courier delivery). This would put the garage door into an open position. At this point the attacker would standby and wait until the garage door is instructed to close. While the garage door is closing the attacker would initiate the jam and capture attack. This would jam the "closed" signal from reaching the MyQ hub that would normally indicate a successful close took place. The MyQ application will then alert the victim that "Something went wrong" and ask for them to try again. This is the third important piece of this attack. Because the application asks the user to try again, they will likely comply as the MyQ hub is saying that the garage door is still open, while the garage door is actually in the closed state, but simply failing to report it to the victim.

When the victim "tries again" to close the garage door from the app, their actions will initiate the garage door opening, since it is already closed. This is when the attacker would jam the now open signal for one full minute or until the new open state transmissions have been successfully jammed. Then the attacker could replay the "closed" signal that was captured during the first close. This will then prompt the user via the MyQ application that the door has been successfully closed while the garage door is actually wide open. This gives the attacker 30 minutes until the remote door sensor does its state sync beacon, which again, could also be jammed. While this scenario is entirely feasible and reproduced in our lab setting, it is much more complex than needed at this point. The attacker can instead simply remove the sensor and place it in a vertical orientation to transmit a true "closed" state to the user via their app.

The only caveat for jamming multiple state sync beacons is that eventually the MyQ hub will say it has lost connection with the remote door sensor. This is only a factor if the attacker wants to be completely covert.

Radio Frequency Attack

In order to selectively jam and capture the signals from the sensor, we make use of an Ettus Research Universal Software Radio Peripheral (USRP) [7] connected to a Linux computer running GNU Radio. Since the original signal from the MyQ is spread over three frequencies, we also need to deal with the same three frequencies. Twice – once for the reception, once for the jamming.

A. Hardware Selection and Setup

We have multiple options to do this (some more realistic than others):

1. Half duplex two-radio frequency hopping: use a single radio to track the frequency hopping of the MyQ and capture a single frequency at a time. Use a second radio which hops in the same sequence and broadcasts a jamming signal. This is an ideal solution, but not particularly realizable.
2. Half duplex six-radio: use three receive mode radios, one per frequency, paired with three transmitters for jamming.
3. Half duplex broad spectrum four-radio: use a broadspectrum capture radio, capture a broad enough bandwidth to encompass all three frequencies, while simultaneously jamming on the three frequencies with three jamming radios.
4. Full duplex three radios: one full duplex radio perfrequency, both capturing and jamming on a single frequency per radio.
5. Full duplex single radio: use a single radio and some filtering to have narrow capture windows across a broad bandwidth, while simultaneously transmitting a filtered broad-spectrum signal to jam all three frequencies.

Other options exist as well but are simply variants of the above. Full duplex requires a more complex radio, but halves the number of radios needed, and therefore the hardware investment.

Each option has its pros and cons. Option 1 requires significant knowledge of the MyQ transmission scheme, which we do not have. Option 2 requires significant hardware resources yet is straightforward and simple in implementation. Option 3 simplifies the hardware load on the receive end but jamming still requires many radios. In our case, option 5 was the best, as we could trade some generation complexity for hardware simplicity – in other words, needing only a single radio and two antennae is worth the extra computation

In order to implement option 5, we use an Ettus Research USRP B205mini-i software defined radio, a pair of VHF/UHF multi-band amateur (ham) radio antennas, and a Linux workstation. An interesting aspect to note is that 312.5MHz is not within the amateur band, and so the antennas are not tuned for it – we are actually using extremely inefficient antennas for our proof-of-concept. Custom tuned antennas would provide significant range and effectiveness boosts.

Figure 2: MyQ state sensor broadcasts valid signals (green) while the USRP broadcasts an on-demand jamming signal (red)

As shown in Figure 2, our setup is with the jamming hardware outside the direct line of sight between the transmitter (MyQ state sensor) and the receiver (MyQ Hub). This replicates a real-world scenario, when the jamming hardware would be typically outside the garage, while the MyQ hardware sits inside. The state sensor broadcasts garage door states (open/closed), which the USRP captures while simultaneously jamming. Figure2 uses green signals for valid transmissions, with red signals representing jamming signals. As all signals are omnidirectional, specific placement isn't particularly critical for this work.

Other full duplex radios would likely work equally well in this context. While the USRP is below $1,000, there has been a surge in hobbyist grade hardware capable of full duplex operation in the sub-$500 range. Options include the LimeSDR [8], the BladeRF [9], and the PlutoSDR [10] – all priced well within the range of someone even casually interested in the area.

B. Multi-spectral receive window

Having decided on a single, full duplex radio, the next question is how to receive on all three frequencies without having to receive everything in between as well.

Figure 3: Band filtering to obtain narrow RX window

Figure 3 shows a snippet of a GNU Radio-companion flowgraph to handle the pass bands of the input signal such that they match the three frequencies we care about. We start with a central frequency matching the central frequency of the MyQ (312.5 MHz), covering 2MHz of bandwidth. As the spectrum of the MyQ uses 311.88MHz, 312.5MHz, and 313.12MHz, 312.5MHz ± 1MHz encompasses all three channels.

In order to filter out the regions between the frequencies of interest, we use consecutive band filters – a low pass filter in the complex domain, and a band reject filter in the complex domain. The low pass filter allows 312.5MHz ± 650 KHz through, meaning any frequencies beyond our upper and lower frequencies of interest are dropped below the noise floor of our receiver.

We couple the output of the low pass filter into a band reject filter to remove the intervening frequencies that we don't want. Specifically, this only allows 312.5MHz ± 35KHz, 313.12 MHz ± 35KHz, and 311.88MHz ± 35KHz to rise above the noise floor of our receiver, giving us the ability to capture all three frequencies from a single antenna. It should be noted

that for our work, it was unnecessary to capture the frequencies into individual files, but our technique generalizes in a way such that we are able to capture individual frequencies each to their own files.

The processing power necessary to filter in this fashion renders this approach difficult on older or embedded hardware – frequent data underruns will occur, meaning the CPU isn't keeping up with I/Q samples from the radio. On more modern hardware, this is a perfectly acceptable tradeoff between computing power and hardware simplicity.

C. Jamming signal generation

Figure 4: Jamming signal generation and broadcast

In the same vein as signal capture, we need to broadcast a jamming signal on three distinct, though related, frequencies. Figure 4 includes the complete jamming signal broadcast path.

We begin with a simple Gaussian noise source (amplitudes are normally distributed) fed into a multiply constant block to provide both amplification (10-fold amplification) and a gating ability – we can disable jamming by setting this value to zero, meaning no noise is broadcast. We once again make use of the low pass and band reject filters as above to create a multispectral signal which mimics the form and relation of the incoming signal.

Rather than broadcasting on the exact frequency as the MyQ state sensor, we shift upwards by 100 KHz. Much like an airhorn preventing you from having a conversation with your neighbor, our jamming signa overloads the MyQ base station receiver, rendering the reception impossible. Our reception window, however, is only 70KHz wide, centered on the frequencies in use. As such, we essentially "don't hear" the air horn that is our jamming signal and can receive the state broadcasts clearly and easily.

D. Signal I/Q capture and replay

Figure 5: I/Q storage and instrumentation sinks

Once the signal has transited the filtering discussed in §IV.B above, we need to store the signal for later replay. This is accomplished via a file sink block in GNU Radio, storing it on the local file system. This is also the point in the flow where we instrument the signal in order to provide visualization. After all, this is spooky action at a distance (aka magic), so some sense of visualization helps us ensure things are working the way we desire.

Figure 6: Signal replay flow

Once we have the file saved, we can finally replay it. Figure 6 shows the simple flow graph necessary for this activity. We simply feed the file we saved earlier and pass it to a USRP Sink, set to transmit on the same center frequency as the MyQ state sensor.

E. Putting it all together

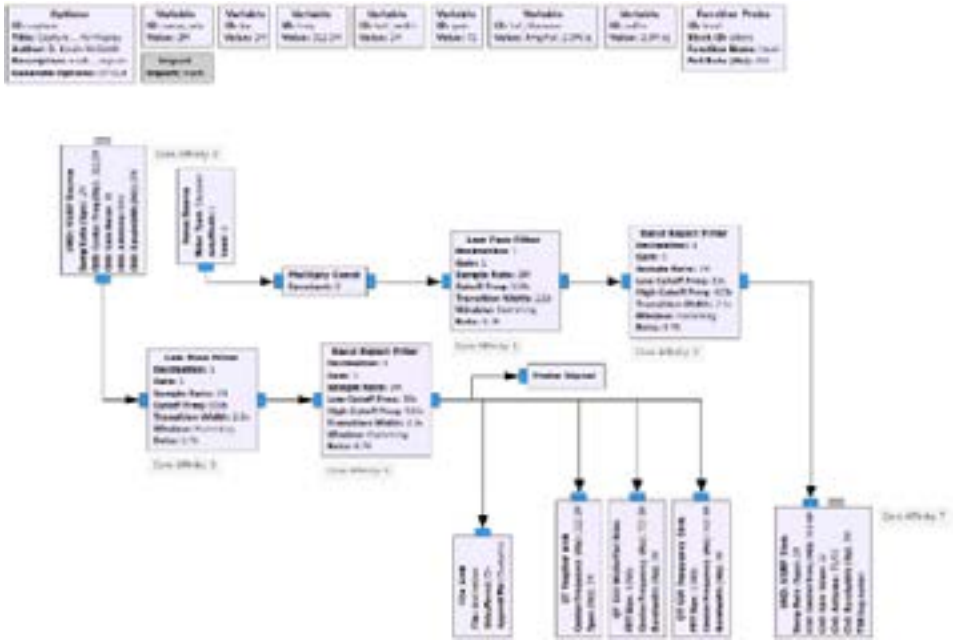

Figure 7: Full Jam/Capture Flowgraph ("Bob")

There is a practical limitation of the USRP – only a single process can be accessing it at any given time. As such, we need to run all the above within the scope of a single process. Figure 8 shows this complete flowgraph, (which we affectionately refer to as "Bob" – because we jammin'), which includes both the jamming and reception flows, as well as the file capture capacity, signal visualization, and configuration variables. The user interface this flow graph generates can be seen in Figure 8.

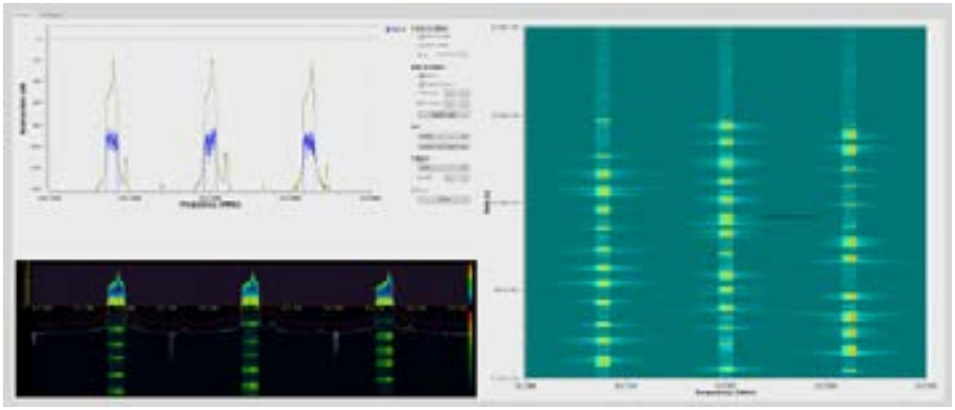

Figure 8: Basic user interface for PoC

While this is the general form of the flowgraph, many small adjustments are necessary when working within the physical world. One of the strangest issues which arose was signal aliasing. Without an appropriately wide bandwidth – 2MHz in our case – we observed the signals multiplying in baffling ways. The root cause of this was analogous to the wrap-around in PacMan – we ran off one edge, and simply popped up on the other. What this means in practice is that we need sufficient bandwidth to encompass both state broadcast and jamming signal, as well as a small buffer to prevent aliasing.

Additional tunable factors include receive gain, transmit gain, and dealing with multipathing as the signal passes through the wall of the garage. These are situational factors which would need tuning at the time of attack, based on the conditions found. In other words, these are some of the engineering steps to fully realize this attack as a black box, generally applicable attack.

F. JIT Jamming

A specific detail to note in Figure 8 is the "Probe Signal" block. This is a GNU Radio sink which allows measurement of the incoming signal. What this means for us is that we can probe the level of the incoming signal from the MyQ state sensor, and level our jamming signal to mimic the strength.

In this fashion, we can sit near the unit, quiescent until such time as the MyQ state sensor begins transmitting. In essence, we enter a stealth mode of operation. One of the drawbacks of this general class of attacks is the jamming signal is detectable, as it usually broadcasts at all times. In our implementation, we are only broadcasting when the valid signals are present, thereby masking our presence.

By using the probe signal to monitor the strength of the signal, we can accurately determine when the transmitter is active. Given the rapid rate of change of the signal, we need to ensure our SDR platform can keep up, and so a reasonably high sample rate is used – 250Hz works well in practice. The on-off time of the USRP platform we make use of is sufficiently small that we can fairly accurately match the active periods of the state sensor.

Potential Mitigations

There are some potential mitigations to this sort of attack. While we present some suggestions below, it should be noted that FCC regulations may preclude either of the RF-based mitigations (A and D below).

A. Hardware 1: Spectrum Separation

The system could move the channels hopped between further apart, outside the receive bandwidth of easy to obtain hardware. This would likely require multiple chips in the final solution, given that the available simultaneous bandwidth of low cost (<$1000) hardware currently reaches nearly 60 MHz, with 120MHz receive windows well within reach of even a moderately funded group. This solution may be infeasible due to FCC spectrum guidelines.

B. Hardware 2: Physical attachment of sensor

The system could use a sensor which is physically connected to the track, sensing when the door arm moves past, and in which direction. This would likely take the form of a pair of inexpensive optical sensors to enable direction tracking, wired directly back to the base station. No RF, no jamming. The problem with this becomes installation process – it's more complex to run a wire than to Velcro a sensor to the door.

C. Software 1: Introduce timing into signal encoding

Currently there doesn't appear to be any sense of time in the broadcast signal. By modifying the software of the base station and transmitter to have synced clocks, as well as a short timeout on the validity of the signals more robust protections could be added against the replay of signals, rendering this attack much more difficult to pull off. Alternatively, a timestamp could be required to be within a certain range of the action

time (e.g. the moment the user initiates an open/close operation) to be considered valid. Lack of a real-time clock would require a new hardware revision on the state transmitter, but the hub could use network time.

D. Software 2: More frequent state checks

A final option to mitigate this attack would be via more frequent state transmissions from the sensor. It's unclear how often would be necessary, but the basic gist is broadcast often enough to shorten the window of access. There are battery life concerns with this solution. It would likely be simpler to implement than option C due to no additional hardware being necessary. This approach would also likely run afoul of FCC regulations due to the frequency of broadcasts.

Conclusion

Software defined radio continues to be an enticing tool for cyber criminals, who are often looking to avoid leaving digital footprints. As radio frequency transmitting devices are often used to physically secure sensitive areas, they are increasingly in need of true security assessments and expert analysis. This paper reflects our research and insight into the type of attack that we believe would be simplistic with some basic hardware. It is essential that vendors continue to embrace and even incentivize the research community and leverage the relationships to understand the attack surface and corresponding mitigations to drive the development of more secure software and hardware.

We'd like to recognize Chamberlain's efforts in both their positive response to this research, demonstrating effective communications, testing and validation, and ongoing mitigation development.

References

1. S. Whyte, "Too Curious For My Own Good," 15 March 2014. [Online]. Available: https://spencerwhyte. blogspot.com/2014/03/delay-attack-jam-interceptand-replay.html?m=1. [Accessed 7 September 2019].

2. S. Kamkar, "Samy Kamkar: Drive It Like You Hacked It,"2015. [Online]. Available: http://samy.pl/defcon2015/. [Accessed 8 September 2019].

3. electronicsnotes, "RF & Microwave Filters: the basics," electronicsnotes, [Online]. Available: https://www.electronics-notes.com/articles/radio/rf-filters/understanding-rf-filters-basics-tutorial.php. [Accessed 16 August 2019].

4. 3200 Creative, "Aliasing of Signals - Identity Theft in the Frequency Domain," AllSignalProcessing, 25 April 2014. [Online]. Available: https://allsignalprocessing.com/2015/04/25/aliasing-of-signals-identity-theft-inthe-frequency-domain/. [Accessed 28 October 2019].

5. National Instruments, "What is I/Q Data?," National Instruments, 12 September 2018. [Online]. Available: http://www.ni.com/tutorial/4805/en/. [Accessed 26 August 2019].

6. M. Q. Kuisma, "I/Q Data for Dummies," Ping Research, 10 April 2017. [Online]. Available: http://whiteboard.ping.se/SDR/IQ. [Accessed 19 August 2019].

7. Ettus Research, "USRP B205mini-i," [Online]. Available: https://www.ettus.com/all-products/usrp-b205mini-i/. [Accessed 10 October 2019].

8. Lime Microsystems, "LimeSDR," [Online]. Available: https://www.crowdsupply.com/lime-micro/limesdr. [Accessed 4 12 2019].

9. Nuand, LLC, "bladeRF 2.0 micro," [Online]. Available: https://www.nuand.com/bladerf-2-0-micro/. [Accessed 4 12 2019].

10. Analog Devices, "ADALM-PLUTO Software-Defined Radio Active Learning Module," [Online]. Available: https://www.analog.com/en/design-center/evaluationhardware-and-software/evaluation-boards-kits/adalm-pluto.html. [Accessed 4 12 2019].

11. Nuand LLC, "bladeRF 2.0 micro xA4," 20 March 2019. [Online]. Available: https://www.nuand.com/product/ bladerf-xa4/. [Accessed 20 March 2019].

12. Osmocom, "gr-fosphor," 20 May 2016. [Online]. Available: https://osmocom.org/projects/sdr/wiki/fosphor. [Accessed 21 August 2019].

13. GnuRadio, "GNU Radio Wiki — Main Page," gnuradio, 20 May 2019. [Online]. Available: https://wiki.gnuradio. org/index.php/Main_Page. [Accessed 25 August 2019].