Security Assessment of Audience Response Systems Using Software Defined Radios

Khai T. Phan
Boston University
Email: kphan95@bu.edu

David Starobinski
Boston University
Email: staro@bu.edu

Liangxiao Xin
Boston University
Email: xlx@bu.edu

***Abstract*—Audience response systems, also known as *clickers*, are used at many academic institutions to offer active learning environments. Since these systems are used to administer graded assignments, and sometimes even exams, it is crucial to assess their security. Our work seeks to exploit and document potential vulnerabilities of clickers. For this purpose, we use software defined radios to perform eavesdropping attacks on an audience response system in production. The results of our study demonstrate that clickers are easily exploitable. We build a prototype and show that it is practically possible to covertly steal answers from a peer or even the entire classroom, with high levels of confidence. As a result of this study, we discourage using clickers for high-stake assessments, unless manufacturers provide proper security protection.**

***Keywords*—Signal processing, clicker, GNU Radio, HackRF One.**

I. Introduction

Many institutions employ Turning Technologies' Response Cards [8], also known as *clickers*, to create active learning environments and encourage students' participation in their classes. Clickers are wireless devices that let instructors poll students for purposes such as taking attendance, and administering quizzes and/or surveys.

Institutions offering classes with clickers include Harvard, Northwestern, Johns Hopkins, University of Colorado at Boulder, Ohio State University, University of California, and Washington State University [7]. Some educational institutions go so far as to administer clicker-based exams. For example, University of Maryland of Baltimore County shows evidence of having administered these types of exams in the past. A post on the university's Division of Information Technology page includes a quotation of a student expressing favor for these exams, commenting "I liked taking the exam on the clickers because we had our own exam booklet in front of us and could go at our own pace. I also liked getting my grade back right away." [9]

The popularity of clickers raises the question of whether these devices are actually secure. In particular, since clickers transmit over radio frequencies, is it possible for a student or another party to sniff answers submitted by other students? By sniffing, we mean the act of eavesdropping communication in order to extract confidential information.

In this paper, we answer this question in the affirmative. We build a prototype of a sniffer using the HackRF One software defined radio platform [4]. Our implementation has an accuracy above 90% if the sniffer is located less than 10 feet away from a clicker and above 70% if the sniffer is less than 20 feet away. Using information provided by the sniffer, a student can cheat in various ways, e.g., by finding out the most commonly submitted answer or by looking at the answer submitted by a particular student (assuming the clicker ID of that student is known).

Our project builds on earlier work by Travis Goodspeed [3] and Taylor Killian [5], who noticed that clickers do not encrypt data while communicating. They also determined the structure of packets sent by clickers. However, their implementation is hardware-based. The attacker uses the nRF24L01/nRF24L01+ radio chips, which are similar to the nRF24LE1 chip used by clickers. Instead, our implementation is software-based.

It is crucial to emphasize that completing such a project with the HackRF One, or any other software defined radio, is a significantly more difficult task. Software defined radios do not offer many of the luxuries which Goodspeed and Killian benefited from, when they worked with the nRF24L01/nRF24L01+ radio chips. Some of these luxuries include *automated demodulation*, which is the act of extracting information from a signal wave, *automated filtering*, which is the process of removing noise or unwanted information from a signal, and *automated error-handling* of abnormal signal behavior. Instead, a software defined radio requires software implementation of the previously mentioned tasks. Despite this downfall, the reward is the versatility of the HackRF One and other software defined radios. They are not device-specific and, once mastered, are modifiable to function with various hardware and signals.

The rest of this paper is organized as follows. We first describe the software and hardware tools used in our implementation. Next, we provide details of the implementation of the sniffer. Then, we present experimental results before concluding the paper.

II. The Tools

A. The HackRF One

The specific software defined radio used in this project to assess the security of Turning Technologies' Response Cards is Scott Gadgets' HackRF One (see Fig. 1). The HackRF One [4] is a hardware device able to capture radio signals via an antenna. It contains a wide-band front end which is able to process frequencies ranging from 1 MHz to 6 GHz. As this range is very large (for comparison, the FM radio broadcast range is only 87.9 MHz to 107.9 MHz), the computational power required to convert between analog and digital signals is expensive. Thus, the HackRF One converts

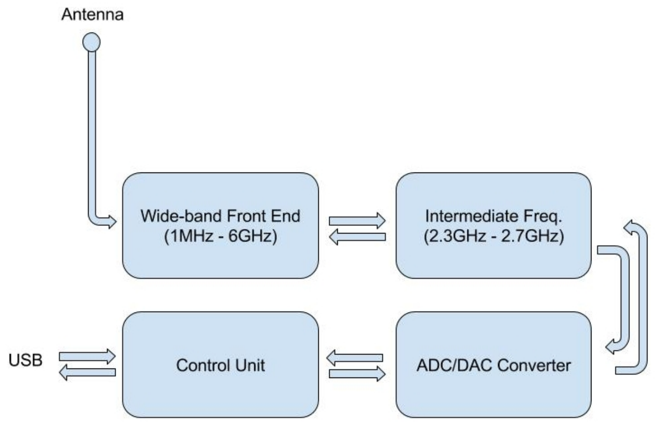

Antenna
Wide-band Front End (1MHz - 6GHz)
Intermediate Freq. (2.3GHz - 2.7GHz)
USB
Control Unit
ADC/DAC Converter

Fig. 1. The diagram above offers a graphical representation, in the form of a flow diagram, of the HackRF One.

this range to an intermediate frequency range of 2.3 GHz to 2.7 GHz. The analog-to-digital/digital-to-analog converter models this received signal as a series of complex numbers. The control unit streams this data into another device, oftentimes a computer operating on a Linux-based operating system. This stream of data can then be modified and analyzed with software. Additionally, a crucial specification of the HackRF is its maximum bandwidth support of 20 MHz.

B. *GNU Radio*

With the HackRF One offering the hardware support for this project, software GNU Radio [2] is used to perform signal processing and analysis on the digital input received via USB port. As mentioned, this digital input is a series of complex numbers. GNU Radio is packaged with a user-friendly interface known as GNU Radio Companion, often abbreviated GRC.

GRC features a variety of graphical tools in the form of building blocks available for the user to construct a circuit-like flow diagram. A complete flow diagram can be used for several purposes, a few being: transmitting signals using a file source or a random value generator, receiving signals and demodulating for sake of playing audio or gathering data, or recording and replaying signals to visually analyze them using one of GRC's many plots. GNU Radio offers the software equivalent of nearly every hardware tool used in signal processing, making it an extremely powerful tool for this project.

C. *Wi-Spy*

Although only briefly used in this project, the Wi-Spy compensates for the HackRF One's comparatively small bandwidth. The Wi-Spy [10] is a USB-compatible 2.4 GHz spectrum analyzer with an internal antenna. Whereas the HackRF One has a maximum bandwidth of 20 MHz, unable to view the entire 2.4 GHz band, the Wi-Spy is able to do so. This is a critical frequency range because the clicker is said to operate within this range (more details regarding the clicker are explained later in the paper).

The Wi-Spy is simple to use; the device operates in conjunction with the Chanalyzer 2.1 software, which launches a graphical representation of the entire 2.4 GHz band, displaying which frequencies on the spectrum contain high amounts of activity. This is especially useful when confirming the operation range of the clicker and selecting a channel within this bandwidth to conduct the product on for optimal results, as channels with consistently high activity affect results by adding excess noise to the signal. Note that this spectrum analyzer does not interact with either GNU Radio or the HackRF One in any way.

III. Implementation

A sniffing attack requires filtering the signal, demodulating it with the correct modulation scheme, synchronizing clocks with the signal's data rate, transforming the demodulated signal into a binary data stream (data consisting of 0 and 1 values), and then interpreting the binary data stream in order to discover packets sent by the clicker. In order to begin implementation of the sniffing attack, it is important to confirm the clicker's specifications found on radio chip manufacturer Nordic's website [6]. Note that the clicker's chip model is the nRF24LE1.

A. *Confirming Operation Frequency*

The first step is to browse through data sheets for specifications regarding the nRF24LE1 radio chip, and then set up tests to either confirm or deny said specifications. The first specification that should be observed is the operation frequency of the clicker. It is crucial to begin with this, as it is impossible to test other specifications if the clicker's signal cannot be located on the frequency spectrum. To ensure that the device does indeed operate in the 2.4 GHz spectrum, the spectrum analyzer Wi-Spy is used. If the clicker operates in this range, an activity spike should occur when the clicker's keypad is pressed. After a brief test period of transmitting signals on several channels of the clicker, it is observed that the clicker does operate in the 2.4 GHz range.

B. *Determining Channel Implementation*

Not only that, but a consistent pattern is also observed during the experiment; each of the clicker's respective channels creates activity on the frequency 2.4*xx* GHz, with *xx* representing the channel number. For example, transmitting a signal on channel 19 creates an activity spike at 2.419 GHz (see Fig. 2).

Additionally, it is found that the minimum channel value is 1 and the maximum channel value is 82, confirmed by both physical attempt and the Federal Communication Commission website [1] (a site that holds the data for many government-approved radio-dependent devices). Knowing this in conjunction with Turning Technologies advertising that the clicker has 82 individual channels, alongside Nordic advertising that the nRF24LE1 operates at either 1 or 2 MHz bandwidth, it can be logically assumed that the clicker operates with 1 MHz bandwidth.

To reiterate what is uncovered thus far, it is now known that the clicker operates in the 2.401 GHz to 2.482 GHz range, with each channel in the 1 to 82 range corresponding to its respective frequency, i.e. channel 40 is 2.440 GHz, and each channel having a 1 MHz bandwidth.

C. *Determining Modulation Scheme*

With knowledge of the clicker's operation frequency, transmission bandwidth, and channel-to-frequency correlation, the next high-priority specification to confirm is the modulation scheme. Although more specifications and assumptions need to be confirmed, it is essential to know how the signals carry data in order to begin implementation.

Thus, the clicker's modulation scheme is discovered via the manufacturer Nordic's website [6] and is found to be GFSK (Gaussian frequency shift keying) modulation, a modulation scheme very similar to FSK (frequency shift keying). This modulation scheme is confirmed by testing the `Quadrature Demod`, a GNU Radio block used for GFSK/FSK modulation, which succcessfully outputs a demodulated signal (see Fig. 3). The significance of the demodulated wave is discussed in the following paragraphs. Now using this knowledge of the clicker's modulation scheme, the clicker's data rate (baud rate) can be tested.

D. *Calculating Baud Rate*

Observing the data sheet once again reveals that the nRF24LE1 radio chip is programmable to either transmit data at a rate of either 1000 kbps or 250 kbps. Also knowing the modulation scheme, operation frequency and bandwidth, and the likelihood of the signal containing a *01010101* 8-bit preamble (both because this is a common preamble and because this is found in the data sheet), the task of discovering the data rate/baud rate becomes less intimidating. Using these pieces of information, the signal can preliminarily be demodulated (see Fig. 3). Because it is assumed that the preamble is a *01010101* sequence, looking at the time stamps of the demodulated signal's peaks (binary 1) and valleys (binary 0) reveal the rate at which data is transmitted, i.e. the baud rate. It can thus be confirmed that the baud rate is 1000 kbps.

E. *Constructing the Implementation*

Now, knowing the baud rate, a GNU Radio software implementation can be created (see Fig. 4). The overall function of this implementation is as follows. The `Osmocom Source` generates a stream of complex numbers based on the signal that the HackRF One receives via its antenna. This stream of

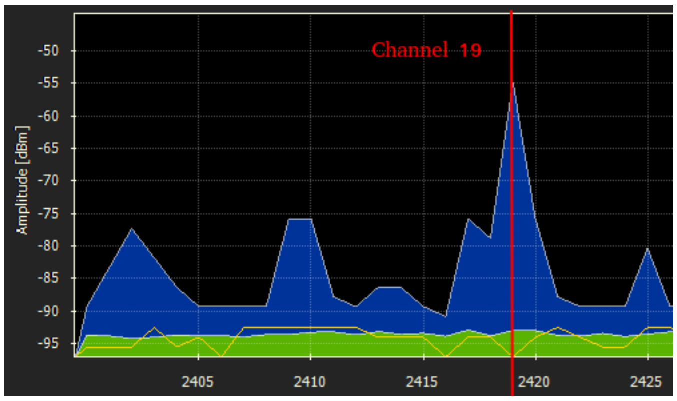


Fig. 2. The screen capture above shows an activity spike at 2.419 GHz resulting from a using the clicker on channel 19.

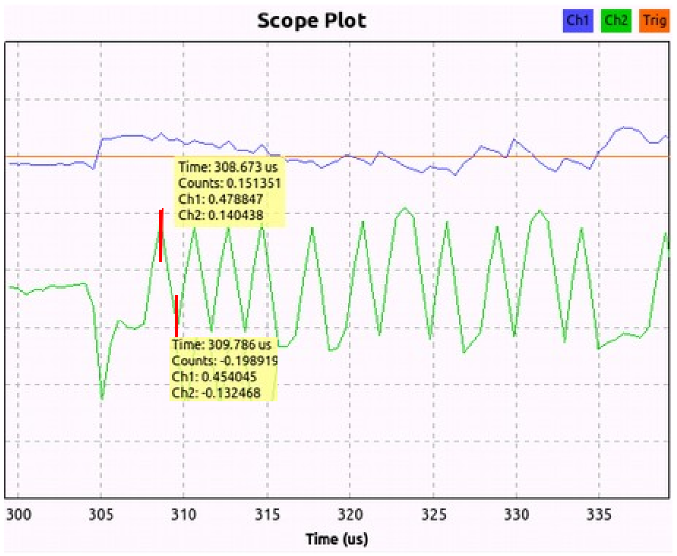


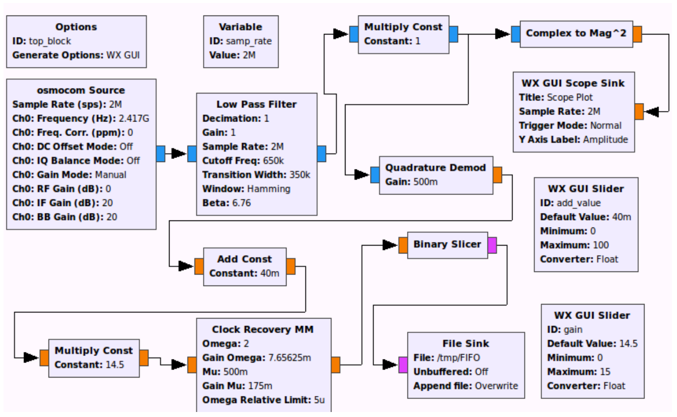
Fig. 3. The screen capture of the demodulated signal. Looking at the peak and the valley, it can be seen that the time between a binary 1 and a binary 0 in the preamble is approximately 1 microsecond. Thus, if it takes 1 microsecond to transmit 1 bit, the baud rate is 1000 kbps.

numbers is passed through a `Low Pass Filter` in order to decrease the amount of excess noise. The `Cutoff Freq` is set to 650 kHz and the `Transition Width` is set to 350 kHz. The combination of these two parameters creates a power decline for signals received outside of $\pm$ 500 kHz range, thus filtering out all signals aside from the desired 1 MHz bandwidth clicker transmission channel.

The filtered stream of data is now passed through a `Complex to Mag2` block and then into a `WX GUI Scope Sink`; the purpose of this is to view when the signal pulses. That same filtered stream of data is also passed through a `Quadrature Demod` block which transforms the modulated signal into a demodulated wave which can be recognized as binary data, where high values (usually peaks) can be considered 1 and low values (usually valleys) can be considered 0. The next four blocks, the `Add Const`, `Multiply Const`, `Clock Recovery MM`, and `Binary Slicer` slightly modify and then decipher the demodulated wave, producing a binary stream.

The `Clock Recovery MM` block syncs clocks with the transmission signal, so that each bit transmitted by the clicker is deciphered properly. Since the clicker's baud rate is previously determined to be 1000 kbps, the `Omega` parameter of this block is calculated to be 2. This calculation is performed by taking the sample rate, 2 mega samples per second, and dividing by the baud rate. 2Msps/1Mbps = 2s/b, i.e. 2 samples per bit. The `Binary Slicer` then selects 0 on the y-axis as a threshold. Since every two samples is a transmitted bit, a clock tick occurs every two samples. Thus, for each clock tick, the bit is recognized as a 1 if the wave is above the threshold and recognized as a 0 if the wave is below the threshold. The `Add Const` and `Multiply Const` blocks improve this process by centering the signal and amplifying it. Lastly, this stream of deciphered bits is pushed into the `File Sink`.

The only remaining step is to create an implementation



```
Options
ID: top_block
Generate Options: WX GUI

Variable
ID: samp_rate
Value: 2M

osmocom Source
Sample Rate (sps): 2M
Ch0: Frequency (Hz): 2.417G
Ch0: Freq. Corr. (ppm): 0
Ch0: DC Offset Mode: Off
Ch0: IQ Balance Mode: Off
Ch0: Gain Mode: Manual
Ch0: RF Gain (dB): 0
Ch0: IF Gain (dB): 20
Ch0: BB Gain (dB): 20

Low Pass Filter
Decimation: 1
Gain: 1
Sample Rate: 2M
Cutoff Freq: 650k
Transition Width: 350k
Window: Hamming
Beta: 6.76

Multiply Const
Constant: 1

Complex to Mag^2

WX GUI Scope Sink
Title: Scope Plot
Sample Rate: 2M
Trigger Mode: Normal
Y Axis Label: Amplitude

Quadrature Demod
Gain: 500m

WX GUI Slider
ID: add_value
Default Value: 40m
Minimum: 0
Maximum: 100
Converter: Float

Add Const
Constant: 40m

Binary Slicer

Multiply Const
Constant: 14.5

Clock Recovery MM
Omega: 2
Gain Omega: 7.65625m
Mu: 500m
Gain Mu: 175m
Omega Relative Limit: 5u

File Sink
File: /tmp/FIFO
Unbuffered: Off
Append file: Overwrite

WX GUI Slider
ID: gain
Default Value: 14.5
Minimum: 0
Maximum: 15
Converter: Float
```

Fig. 4. GNU Radio implementation of the sniffer.

TABLE I. SNIFFING BENCHMARKS

Distance	1 ft	5 ft	10 ft	15 ft	20 ft	25 ft
Accuracy	97%	98%	92.5%	77%	70%	50%

which works in real time, i.e. while the program is running. The solution to creating a sniffer that runs in real time is using a pipe. With the current implementation that can write the binary data of packets to a file, using a file that can be simultaneously written to and read from is the most intuitive option. Thus, the GNU Radio flow diagram for the implementation remains the same as the secondary implementation. The only difference is that the flow diagram now outputs into a file known as a FIFO. A FIFO is a method of piping data in Linux operating systems. It is a file which requires both a read and a write action operating on it simultaneously. When that condition is met, the FIFO acts as a queue: first in, first out. Binary data is pushed into this FIFO by GNU Radio and the HackRF while a script reads and parses through the file, deciphering it for packets, resulting in a sniffer that operates in real time.

IV. BENCHMARKING AND ANALYSIS

For the sniffing attack, the benchmarking process is rather intuitive. A well-functioning sniffer is one that has a high accuracy of sniffing packets at great distances. Thus, the scenario used to measure accuracy is as follows. The HackRF One, i.e. the sniffer, is positioned at varying distances away from the clicker. The clicker is then set to transmit a packet. If the HackRF One receives and deciphers this packet correctly (ID and payload must be correct), then the packet is considered successfully sniffed. This experiment is performed at varying distances, using two hundred trials for each respective distance. The results of this benchmarking is displayed in Table 1.

To begin analysis, the goal of sniffing is to stealthily and passively acquire knowledge of others' answers and packet submissions. According to benchmarking results, sniffing performs quite well within a lecture hall or classroom setting. Distances of 1 to 10 feet have an accuracy higher than 90 percent and distances 15 to 20 feet have an accuracy higher than 70 percent. The accuracy drops below 50 percent at a distance of 25 feet away from other clickers. We note that in most scenarios, the user would be sitting near other clickers, generally within a vicinity of 25 feet radius. Thus, the sniffing attack is expected to receive most answers that are submitted within the classroom if the sniffer is positioned near the center of the classroom or lecture hall.

V. CONCLUSION

Given that the performance of the sniffing attack are found to be practical and usable within classroom environments, it is reasonable to discourage the use of the clickers for high-stake assignments in classrooms and lecture halls. Although it is safe to use these clickers for surveying and participation purposes, the use of such devices for graded assignments bears the risks of cheating within a class. This is even more true given that the sniffer is found to work for all generations of the clicker. Note that this does not apply only to clickers; it is reasonable to conclude that using any audience response system for graded assignments without knowledge of its security implementations bears the risks of cheating.

Furthermore, the usage of software defined radio is gaining popularity. Because of this, it is important to stay cautious when using any radio-based devices. The clicker is just one of many potentially exploitable devices. Wireless keyboards and mouses can be popular targets as well, especially since it is possible to sniff key strokes and mouse movements while the user is operating with confidential information such as paying bills, checking their bank balance, or logging into their email account. In this new era of technology, it is always good practice to ensure devices and programs are secure by checking their security implementations.

ACKNOWLEDGMENTS

This work was supported in part by NSF under grants 1012910 and 1409053.

REFERENCES

[1] *Federal Communications Commission*. URL: https://www.fcc.gov/.

[2] *GNU Radio*. URL: http://gnuradio.org/.

[3] Travis Goodspeed. *Travis Goodspeed's Blog*. July 2010. URL: http://travisgoodspeed.blogspot.com/2010/07/reversing-rf-clicker.html.

[4] *HackRF One*. URL: https://greatscottgadgets.com/hackrf/.

[5] Taylor Killian. *Taylor Killian*. Nov. 2012. URL: http://www.taylorkillian.com/2012/11/turning-point-clicker-emulation-with.html.

[6] *nRF24LE1*. URL: http://www.nordicsemi.com/eng/products/2.4ghz-rf/nrf24le1.

[7] Fahmida Y Rashid. *The Rise of Clickers Is Starting to Change How College Professors Run their Classrooms*. Jan. 2011. URL: http://www.villagevoice.com/news/the-rise-of-clickers-is-starting-to-change-how-college-professors-run-their-classrooms-6429859.

[8] *ResponseCard RF*. URL: https://www.turningtechnologies.com/response-solutions/responsecard-rf.

[9] *Students More Accepting of Using Clickers for Exams*. Apr. 2014. URL: http://my.umbc.edu/groups/doit/posts/44012.

[10] *Wi-Spy*. URL: http://www.wi-spy.co.uk/index.php/products/wi-spy.