Replay Attack Using Software Defined Radio

| James Cogswell  Cyber Security Engineering  George Mason University  Fairfax, VA  jcogswe@gmu.edu | Abdulazeiz Almarshoodi  Cyber Security Engineering  George Mason University  Fairfax, VA  aalmarsh@gmu.edu | Ryker Gogolkiewicz  Cyber Security Engineering  George Mason University  Fairfax, VA  rgogolki@gmu.edu |
| --- | --- | --- |
|  |  |  |
| Andrew McConnell  Cyber Security Engineering  George Mason University  Fairfax, VA  amcconn@gmu.edu | Nicholas Visone  Cyber Security Engineering  George Mason University  Fairfax, VA  nvisone@gmu.edu | William Yuan  Cyber Security Engineering  George Mason University  Fairfax, VA  wyuan2@gmu.edu |

**ABSTRACT**

In modern society, it is commonly believed that technology we interact with daily such as vehicles, garage doors, and any other device that communicates using radio frequencies (RF), are not susceptible to relatively trivial attacks by hackers with $300 devices. The reality is that even vehicles made in the last year have been susceptible to replay attacks, allowing the actor to lock & unlock the doors, or even start the engine–with nothing but a $300 software defined radio (SDR), and a computer or Raspberry Pi with software that can interface with the SDR. Attackers can search for frequencies of various devices on the FCC website and prepare their SDR to record communication on that frequency. When the owner presses the button to unlock their car door or open their garage, the attacker can potentially capture that recording to unlock the vehicle or open the garage door when the owner leaves. This paper examines software defined radio being used to perform a replay attack. Some devices are harder to attack than others and while these more sophisticated attacks will be explained, the attack demonstrated in this paper will be performed on a device that is vulnerable to a relatively basic replay attack. The goal with this paper is for readers to gain a basic understanding of the risks associated with this type of RF attack.

**KEYWORDS**

software defined radio, replay attack, rf attack, hackrf one, security

**1 OUTLINE**

This project will be performed using a Great Scott Gadgets HackRF One SDR peripheral which has a frequency range of 1MHz to 6GHz. The Universal Radio Hacker application is installed on an Ubuntu Linux virtual machine which will be used to enable the HackRF to capture the frequency of the source device and transmit–or *replay*–the same frequency to achieve the same result as if the source device had been activated again. Examples of this include unlocking a car door, opening a garage door, or ringing a wireless doorbell, by performing a replay attack. This project will provide an overview of RF technology, highlight common attack vectors associated with RF communication devices, and introduce a basic replay attack using a fairly affordable SDR device and free software.

**2 AN ARCHITECTURE FOR SECURE SOFTWARE DEFINED RADIO**

This paper goes over topics including software and how it relates to radio devices [1]. There are several vulnerabilities related to radio and this article goes over how to mitigate them. There is a hardware vulnerability involving the driver for hardware. An operating system separates the driver from the hardware and prevents malicious code from being injected. Also, there is a method of changing the process ID once a malicious process is discovered. Finally, another vulnerability is when data with a structure is passed into the V-HW with illegal parameters. Illegal parameters can include malicious code, therefore the system executes the code and damage is done. Security policies can be implemented to check for these specific instances.

**2.1 Comparative Summary**

The technique used in this paper was a software defined radio (GNU Radio) running on a Linux virtual machine. One advantage of this paper is this is almost exactly how our demo was performed. Except, the software was run through the main Windows OS and we used Universal Radio Hacker. One advantage learned from this paper is that overhead (in our case noise) can be eliminated by decreasing the amount of overhead. By using a virtual machine, this eliminates overhead by requiring the packet capture to go through an isolated virtual ethernet connection. This will allow for less noise when capturing the digital signal from the dip switch (garage door opener). When running on the main OS, there is a risk of noise. The amount of processing power required to analyze the captured signal is not much, so our laptops were more than capable of handling the software to analyze. Another disadvantage is we do not know what security policy is in use and if this has any effect when capturing the signal. If we were able to identify the security policy being used we could modify it and capture a better signal. Due to the nature of waves, we received too much noise possibly from other things in the house since we captured the signal indoors. These things could include a TV, microwave, sound system, etc. One advantage of using Universal Radio Hacker is that it is very easy to set the sampling rate and filter through patterns of 0s and 1s. This is very important since we are trying to find the Off and On pattern of the signal.

**3 HIJACKING UNMANNED AERIAL VEHICLE BY EXPLOITING CIVIL GPS VULNERABILITIES USING SOFTWARE-DEFINED RADIO**

This paper describes and conducts 3 experiments with UAVs - force drone to land, force drone to fly to an attacker-specified area, and force drone to land in that area [2]. Paper describes that the combination of civilian GPS being unsecured and SDR becoming more accessible to civilians, it is incredibly easy to launch attacks against UAVs. The HackRF is not suitable alone in attacks against GPS based on its low accuracy oscillator, so they use something called a temperature compensated crystal oscillator (TCXO). Basic research into a 0.5 ppm tolerance TXCO reveals that it can sell for about $20-$60 meaning the buy-in for GPS spoofing attacks is around $350, assuming you have a computer.

Stated in the background section, there are 2 bands for GPS, civilian at 1575.42 MHz, and military at 1227.60 MHz. The military band is encrypted while the civilian one is unencrypted. On top of being unencrypted, the civilian GPS’s spreading codes are widely documented and available on the Internet, meaning that you can easily research a captured signal.

The techniques the authors used were spread across 3 kinds of attacks. For the first kind of attack, they tried to make the drone land. They had an advantage for this attack method, being that drones will usually land at their current location if they lose messages from the remote controller. They would transmit a fake GPS signal to the drone which would trick it into thinking that it was in a no-fly zone. The drone will then fly out of the area then land. The second attack was to make the drone fly in a direction specified by the attackers. For this attack the authors took advantage of a vulnerability in the stable hovering mechanism. This helps make the drone stay still while flying and not receiving messages from the remote control. This is achieved by recording the location where it should stay, and it will adjust its location to stay in that original location. The attack works to change the recorded location and make the drone fly to a ‘spoofed’ location chosen by the attackers. For the final attack method, the authors wanted to make the drone land in an area specified by them. This is a combination of attacks 1 and 2. It utilizes the go-home function of drones which makes the drone go back to its take off point. It would then feed the drone spoofed GPS signals to go to a point specified by the attackers/authors.

**3.1 Comparative Summary**

Some advantages that we can use for our project are these attacks are outlined as flow charts on the paper. These charts are how they conducted each step of a certain attack. The disadvantage is that we can only really draw insights as a drone has exponentially more functions than a garage door, especially one without rolling codes.

While out of scope for this project, it shows that what we are planning on doing is translatable to larger scale attacks at just another ~$50 dollars away (on top of the HackRF). While there are three different attacks and avenues, each with increased difficulty compared to our project, this drone exploitation project relates a lot to attacks on modern garage doors. Modern cars and garage doors open/close or lock/unlock with a rolling code system, which changes the code every time. There are attacks involving capturing the signal, replaying it to unlock the device at the same time the user clicks the button again, and then capturing that second signal to use later. In short, the charts of their attack patterns can help us to structure our own attacks, but the complexity of this paper’s methods dwarf ours by a large amount.

**4 SECURITY ASPECTS IN SOFTWARE DEFINED RADIO AND COGNITIVE RADIO NETWORKS: A SURVEY AND A WAY AHEAD**

“Security Aspects in Software Defined Radio and Cognitive Radio Networks:A survey and a Way ahead” [3], article talks about the future threats and protection techniques that Software Defined Radio face. Along with Security requirements that are needed SDR. The NSA has incorporated a strict list of rules for the security requirements. “Measures intended to protect and defend information and information systems by ensuring their availability, integrity, authentication, confidentiality, and non-repudiation.” The rules are set to protect against two main security issues with SDR’s which is, “Who guarantees that the downloaded profile or software module comes from a trusted source and can be activated on the SDR device? And Who guarantees that the downloaded profile or software module will behave as expected. These two questions pertain to the possibility that attackers may be able to intercept the SDR through wireless connections as they are similar to wireless systems. Attackers may be able to download malicious software that could take over the SDR affecting wireless networks. Some of the threats pertain to Insertion of malicious software, alteration or destruction of the configuration data, unauthorized use of software defined data to just name a few.

**4.1 Comparative Summary**

The techniques in “Security Aspects in Software Defined Radio and Cognitive Radio networks:A Survey and a Way Ahead” describes Software Defined Radio (SDR) and Cognitive Radio (CR) on their applicational usages along with security measurements and countermeasures for SDR and CR.This article gives a very good and detailed explanation on different types of defensive security measurements such as Protection of confidentiality, protection of integrity, accountability and many more.

Comparing the article to our project on working with hackRF garage doors and manipulating on what can be done through opening the door and applying a replay attack. The article itself gives a good indication of what could be done to prevent hackRF attacks. The constraints that are provided on SDR and CR are not as applicable to our project itself, due to us not having any constraints on our hackRF garage door opening.

**5 OVERVIEW OF ISM BANDS AND SOFTWARE DEFINED RADIO**

This article, “Overview of ISM Bands and Software-Defined Radio” [4], describes frequency bands of popular frequencies especially for home automation. There is some testing done to find the signals from a wireless remote for an electric switch using HackRF. It shows that the 433 MHz signal is captured from the electric switch using a software called “SDRSharp”. It shows how the software is used to show the on and off signal. After capturing the signal, the author then uses the program “GNU Radio” to process the signal. Overall, this article discusses the different frequency bands that are used frequently and describes where certain bands are used in different ITU regions [4].

The software “SDRSharp” that is used in the SDR experimentation is a free and popular software to be used with RTL-SDR [4]. There are multiple configurations to be set with the software to be optimized for use which sometimes can be hard to understand what to set the configuration at. Since the manual does not go into depth into each setting and there are not that many videos that use this software for a replay attack.

**5.1 Comparative Summary**

The research that was mostly done in the article was about the different frequencies that are popular for home automation devices. The research used the tool “SDRSharp” to scan for the 433 MHz frequency which was used to capture an electric wireless socket that has more than 100ft range [4]. The author then uses the program “GNU Radio” to process the blocks of the signal where the author shows the FFT block.

The advantages of this article is that it explains some of the theory part regarding the signal and on and off keying. It includes the functions that represent the OOK.

The disadvantage of this article is that it does not go into detail into each software, it just goes through the result and ignores the process. It is hard to redo the experiment with just this article.

Compared to the experiment our team did, the process that our group used was much easier with Universal Radio Hacker. The program “SDR Sharp” required more setup and configuration to capture the signal where the URH just required us to set up the frequency where we found the frequency by looking up the ID of the device on FCCID.io and attach the HackRF to the software and then start the signal capturing process. The process was minimal without any complications. There are some compatibility issues that are known to happen with “SDR#”.

**6 EXPLOITING BUFFER OVERFLOW VULNERABILITIES IN SOFTWARE DEFINED RADIOS**

This paper expands on the idea of the previously known buffer overflow vulnerability in the processes of programs which allow for the attacker to overwrite the adjacent memory locations while writing data to a buffer and presents the possibility of this exploit being able to modify behaviors of waveforms in software defined radios [5]. The two specific vulnerabilities explored in this article are heap overflow exploits and stack overflow exploits. In a heap overflow attack, the framing protocol of the waveform is exploited. The waveform is vulnerable to attackers due to the framing protocol making the assumption the length of the frame will never surpass the maximum length allowed. When an attacker is able to insert data longer than the maximum length, the frame synchronizer block, used to search for and validate the payloads, continues to search for the end flag and save incoming bits. In the stack overflow exploit, stack smashing was used to cause a buffer overflow and corrupt the memory locations within the stack frame and result in a denial-of-service. The shift from hardware to software firmware provides new security challenges and due to the much quicker development life cycles of software based radios, these vulnerabilities are often overlooked and exploitable.

**6.1 Comparative Summary**

The method used in the research paper was to attack the stack and memory address locations of the systems using software-defined radio. In our project, we are recording the radio signal, cleaning it up, and performing a replay attack without directly attacking the stack or memory. These two methods of attack are very different and replay attacks rely on reconnaissance to discover the frequency required to perform the replays. Buffer overflow attacks are invasive and intrude on the system itself to negatively affect it with the potential of memory corruption, denial of service, and escalation of privileges.

Reading this article did provide insight into the architecture of software-defined radio devices and the vulnerabilities associated with the devices. While not entirely relevant, this information serves as providing a good background understanding on the ins and outs of the devices associated with software-defined radio and how as systems become more dependent on it, the vulnerabilities must be addressed.

**7 DEVELOPMENT AND TESTING OF A REAL-TIME LoRaWAN SNIFFER BASED ON GNU-RADIO**

This paper describes the vulnerabilities which exist in a wireless sensor network over a long-range wide area network, or LoRaWAN. Network attacks used in their research utilized a protocol analyzer (sniffer) to capture packets sent through the network. To sniff the packets, the hardware RTL2832U was used along with GNU-Radio to define the parameters which the RTL device would sniff, and then Wireshark was used to visualize the sniffed data. A HackRF One SDR device and GNU-Radio were used to perform a replay attack on the LoRa server verification to show that availability and confidentiality of the data could be threatened [6].

LoRaWAN has two node activation processes: 1) Over The Air Activation (OTAA), where a node sends a join request to the gateway, which responds with the data from the network server; and 2) Activation by Personalization (ABP), where required information is stored in the nodes, so communication is not required for access to the network. Additionally, LoRaWAN networks have two security layers: 1) the *Network Session Key*, to ensure authenticity of the node; and 2) the *Application Session Key* to ensure data reliability. In their research however, some attacks were performed on a Class-A network in order to test the security provided by the protocol [6].

To perform this attack, they first eavesdropped on the network to passive capture network activity using GNU-Radio, Wireshark, and the RLT2832U. Four Class-A nodes were configured in a star topology, in which the gateway and the sniffer receive the sent data [6]. Next, they performed a replay attack by copying the transmitted signal to supersede the node by using the HackRF One, which was configured in GNU-Radio to allow it to make copies of the transmissions to send them later. The replay tests used three different configurations: 1) verify the server accepted the data copied from the node 2) copy the data without the join request 3) utilize a frame counter [6].

Results from the eavesdropping portion of this attack were successful, but the sniffer did not notify Wireshark of the response messages from the server to the nodes. Additionally, the frame counter was not encrypted.

Results from the replay portion of the attack are described for each of the three aforementioned configurations used. For the first configuration, success verification was verified by inspecting the application response of a message sent by a legitimate node and comparing it to the application response of a message sent by the malicious node. The node’s request messages and the server’s answer messages are required because they are captured by the malicious device which copied the signals. But, the answer messages are irrelevant in this test where the attack is continuously performed since the request and answer messages are sent again by the malicious node, causing the attack to fail. Therefore, even if the response messages from the server were removed from the copied file, the attack would still fail due to the inability for the malicious device to obtain authentication keys sent in the response to the faux join request [6].

As for the second configuration of the replay portion of the attack, success could not be found. Even though data sent by the attacker node was received by the gateway and the identifier matches that of the original node, the server rejects the data due to a synchronism failure in the message’s counter field. Whether or not the legitimate node continues sending traffic, the counter in the attacker node will never match the number stored in the counter that controls the server [6].

The third configuration was also unsuccessful. As mentioned previously, the authors stated that “the success or failure of the attack is verified by comparing the responses of the application to the different messages that were sent” [6]. The attack fails once again due to synchronization issues as well as malicious nodes sending messages before the authentic node. The server only accepted messages from the legitimate node, and rejected all messages from the attacker node. Activation by ABP contains a critical vulnerability since the keys are invariable and have no need for constant authentication in the network. So theoretically, so long as a malicious message meets certain requirements, it can be accepted by the LoRaWAN network server. Although these criteria were met, the attack was unsuccessful because initialization of the nodes was carried out by OTAA. This creates new keys every time a new session is concluded [6].

Further issues in their attack were present as well, such as the inaccuracy in copied signals by the HackRF One due to interference, sampling rate, or message transmission power. Because of this, the message format was invalid for the application and consequently rejected, despite the message matching the value of the counter [6].

**7.1 Comparative Summary**

Although the research conducted by Bravo-Montoya, et al. is largely focused on network attacks, their approach is very similar to ours in some ways. The researchers used HackRF One SDR and GNU-Radio software. We had originally planned to use GNU-Radio software but compatibility issues forced us to choose something else, hence the choice to use Universal Radio Hacker. Where our project differs the most from their research is that we are only using one SDR device ⎼ the Hack RF One ⎼ for the entire project while they utilized RTL2832U SDR for sniffing network packets and the HackRF One for replaying malicious traffic. Without the additional equipment (such as a second SDR), we were unable to perform many of the attacks that circumvent the security controls built into the vehicle key fobs, and therefore chose another RF device that also highlights the very real security concerns involving someone’s personal property by performing our attack on a home garage door.

Advantages of Bravo-Montoya et al.'s research include access to more equipment and the ability to establish their own network in which they used to test their attack ⎼ as opposed to having to find a network that has the exact components and architecture they believe will produce the best results. However, their research topic seems to be far more complicated than ours.

Disadvantages of their research mostly seem to be centered around the fact that their attack was unsuccessful due to existing protections. Bravo-Montoya et al. speculated that given a few changes to the architecture and components it could be successful, but tests against their assumptions were not done.

Our research and hands-on project work have included many ups and downs, but the overall goal is to examine the process of successfully attacking an RF device that controls entry into either a vehicle or a home. Like Bravo-Montoya et al.’s research, we were unsuccessful at our attempt to perform a replay attack against a vehicle’s key fob with the intention of gaining access to the vehicle. However, after choosing the garage door ⎼ which is probably more alarming ⎼ we were able to find one that was susceptible to this attack since it doesn’t use authentication such as rolling codes.

**8 PROGRESS REPORT**

At this point, many hours have been spent by the group to successfully perform a replay attack on a RF device. The goal is ultimately to utilize the HackRF One SDR as our receiver and transmitter to successfully replay a RF signal to get the receiver device to respond as though the original legitimate remote was the sender. The stretch goal was to perform the attack on a vehicle, but the authentication in all vehicles that the attack was tested on were too difficult to overcome. It is evident that RF replay attacks on vehicle key fobs are far more difficult today than they were a few decades ago.

A backup plan was developed in case the attack on vehicles proved to be outside of the group’s technical capabilities. In that case, the group would perform an attack on a group member’s garage door, which uses DIP switches to define a static key as opposed to rolling codes. One group member confirmed their garage door to have a remote that uses DIP switches and the attack was carried out successfully.

Using the HackRF One SDR along with Universal Radio Hacker software, the group was able to successfully sniff the transmission from the garage door remote, interpret and analyze the signal, and then generate a new, clean signal to be used in the replay attack. After much trial and error, the newly generated signal successfully opened the garage door.

This attack was considered to be sufficient for our topic since it was not trivial for the group’s skill level and it takes an experienced professional to even have a chance at attacking a vehicle’s key fob. However, the same attack the group performed is the foundation of an attack on any RF device. To attack a device with preventative measures built in, the attacker would go through the same process as our group did, but will need to change aspects of the attack in order to overcome the security measures.

**REFERENCES**

[1] Li, C., Raghunathan, A., & Jha, N. K. (2009). An architecture for secure software defined radio. *Automation Test in Europe Conference Exhibition 2009 Design*, 448–453. https://doi.org/10.1109/DATE.2009.5090707

[2] Zheng, Xian-Chun and Sun, Hung-Min, Hijacking Unmanned Aerial Vehicle by Exploiting Civil GPS Vulnerabilities Using Software-defined Radio, Sens. Mater., Vol. 32, No. 8, 2020, p. 2729-2743. <https://doi.org/10.18494/SAM.2020.2783>

[3] G. Baldini, T. Sturman, A. R. Biswas, R. Leschhorn, G. Godor and M. Street, Security Aspects in Software Defined Radio and Cognitive Radio Networks: A Survey and A Way Ahead, in *IEEE Communications Surveys & Tutorials*, vol. 14, no. 2, pp. 355-379, Second Quarter 2012, doi: 10.1109/SURV.2011.032511.00097.

[4] Kumbhar, A. (2017). Overview of ISM Bands and Software-Defined Radio Experimentation. *Wireless Personal Communications*, *97*(3), 3743–3756. https://doi.org/10.1007/s11277-017-4696-z

[5] S. D. Hitefield, M. Fowler and T. C. Clancy, "Exploiting Buffer Overflow Vulnerabilities in Software Defined Radios," *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, 2018, pp. 1921-1927, doi: 10.1109/Cybermatics\_2018.2018.00318

[6] Bravo-Montoya, A. F., Rondón-Sanabria, J. S., & Gaona-García, E. E. (2019). Development and Testing of a Real-Time LoRawan Sniffer Based on GNU-Radio. Tecno - Lógicas (Instituto Tecnológico Metropolitano), 22(46), 185–194. https://doi.org/10.22430/22565337.1491

###### Individual contribution report:

The team was very responsive and professional. All team members contributed equally and have put effort into learning and executing the experiment. All efforts including research into the topic (i.e., reading research papers), writing the paper for each of the first two phases, researching the HackRF One and Universal Radio Hacker, and performing the actual attack were all done together as a group with equal input from each member. The team had a meeting on 11/6/2021 at 10 am till around 2 pm discussing and analyzing the data after capturing the signal and successfully replaying the signal to open and close the garage door. The team was able to stay on time by completing the experiment.