BU College of Engineering

Wireless Security: From 1s and 0s to invisible espionage threat

EC-521 (A) - Group Project (Fall 2021)

Students in the team:

- Harshit Agrawal - harshit@bu.edu
- Yasmin Almousa - yasminm@bu.edu
- Piyusha Dongre - pddongre@bu.edu

What is the problem that we are trying to solve? Why is it important?

Nowadays, we're using thousands of wireless devices and there are 100s of them that transmit information without proper security protocols in place. Firewalls and rules are in place, intrusion detection systems are running, and all industry best practices are being followed. Great, but what about Radio Frequency attacks? They nimbly sidestep all of that.

It's that variety of different ways in which RF can be used that make it important for security professionals to understand something of the basics of radio. In the past, it was all about how to get an RJ45 connection to a network. Today, it is all about intercepting radio signals such as Bluetooth, Wi-Fi, 4G, and now 5G. Once transmitted into space, a radio signal can be intercepted by anyone with a receiver tuned to the proper frequency. Building or buying receivers for sniffing is easy, and the new technology market is making it more accessible. We'll be looking at wireless vulnerabilities which commonly exist in IoT Devices, and possible ways to mitigate them.

Problem Statement

In this project, we explore various possible ways of a Replay attack and possible prevention for that. There are quite a number of wireless devices that are transmitting information without proper security protocols in place. Once transmitted into space, a radio signal can be intercepted by anyone with a receiver tuned to the proper frequency.

Signal recording and replaying are widely used in data analysis and testing. Our project focuses on wireless vulnerabilities which commonly exist in IoT Devices and possible ways to mitigate them using a cost-effective record-and-replay prototype based on RTL-SDR. We designed and developed a system to sniff data packets for various commercially used IoT devices and sampled these signals to output the raw analog signals again to perform a hijacking attack.

Introduction to Replay Attack

In the world of security, there are innumerable methods of compromising a network. Whether the attacker's goal is to shut down communication between parties, or to steal information from an unsuspecting victim, the security system in place must be able to defend against it. The best method of protecting information, which prevents a lot of these attacks from being feasible, is to encrypt all data that is transmitted from one place to another. However, what happens when the attacker doesn't need to know what the message actually says? This is the principle behind the Replay Attack.

Replay Attacks are listed as entry 294 in the Common Weakness Enumeration (CWE), where they are described as a "flaw [that] exists when the design of the software makes it possible for a malicious user to sniff network traffic and bypass authentication by replaying it to the server in question to the same effect as the original message" (Mitre, 2015). Bypassing authentication, in this case, could mean sniffing for hashed passwords and replaying them back at the server, still encrypted, as was the case with CVE-2005-3435 and CVE-2007-4961. In both of these cases, hashed passwords were simply played back to the server, and entry was allowed.

A replay attack is a general term used to describe a plethora of attack methods. Because there are so many ways to perform such an attack, there is no single way to prevent them. Breaking them down into different classes of attack, though, make it much easier to identify how an attack method is being implemented, and defenses specific to that class of attack can be applied. Paul Syverson, in his paper "A Taxonomy of Replay Attacks", [1] outlines a thorough classification structure for replay attacks. The key concept to recognize is that replay attacks prey on both parties in communication, so attack methods must first be separated into Origin and Destination.

External and Internal Replay Attack

The Origin of an attack can be either internal or external to the running process. An **external replay attack** occurs when a message from outside the current communication is used. Kamkar's RollJam [2] is an example of a "run external" attack. The message used for the attack came from a previous transaction between key and car. That old message was then replayed in a separate transaction to carry out the attack.

This is different from an **internal attack**, where the message being replayed is part of that same stream of communication. Consider a Man in the Middle (MitM) attack. In a MitM attack, a malicious third party hijacks the communication channel between two parties. All traffic goes through them, and they can choose to send, modify, or discard any packets they want. If, for example, a person were trying to withdraw money from their bank account, an attacker could identify the critical message, change the bank account number, and replay the modified signal as many times as they wanted. Even if there are layers of encryption protecting the identity of the victim, the bank server will still recognize the transaction as valid because the attacker doesn't need to know who they are stealing from.

The Destination of an attack can either be deflected or sent straight through. Both aforementioned attacks are examples of straight replays. The intended target was also the target for the attack. The adversary simply delayed, duplicated, or modified the original package to conform to their own purposes. Deflection covers any message that is redirected away from the intended source. There are two possibilities for who could receive the message. First, the signal could be replayed to a third party. If a message from a server is received en route to a user, an attacker could decode the message and possibly create fake messages to trick other users into thinking they are the server. If the message isn't being deflected to a third party, and it isn't being forwarded to the second party, then that only leaves reflection back to the sender. The full taxonomy can be put thusly:

1. Run external attacks
    a. Interleavings
        i. Deflections
            1. Reflections to sender
            2. Deflections to the third party
        ii. Straight replays
    b. Classic Replays
        i. Deflections
            1. Reflections to sender
            2. Deflections to the third party
        ii. Straight replays
2. Run internal
    a. Deflections
        i. Reflections to sender
        ii. Deflections to the third party
    b. Straight replays

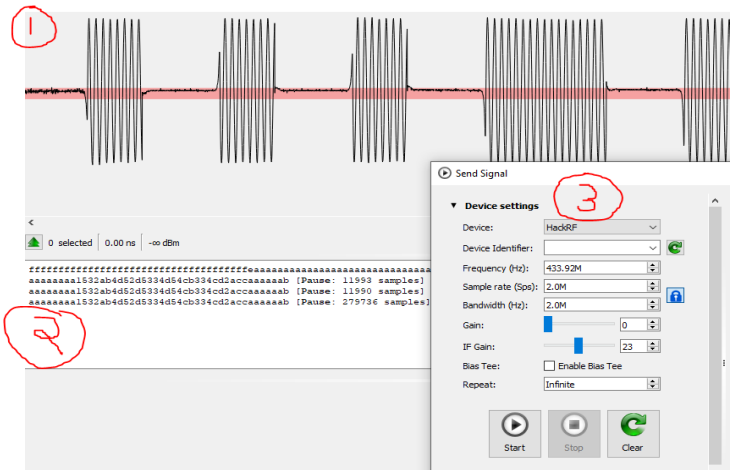
Interleaving and Classic Replays describe methods of attack that are either message from one process being injected concurrently into another process (interleaving) or attacks that don't depend on the time of the session (classic). While delineating attack methods involving these classes is out of the scope of this paper, as they begin to employ much more complex authentication strategies, it is important to note their existence when observing the strengths and weaknesses of different defenses.

Capturing Signal and Inspection

The goal is to make as many RF captures in various conditions (on, off, constant on, specific flash patterns, etc.) as possible. This is essential since diverse samples will allow us to compare different packets and reveal the overall packet structure. Especially when looking at real RF systems, taking as many captures as possible under all pertinent conditions is an important step. It's reasonable to assume that a packet's content determines the car door lock/unlock, or room lights/fan control. In real systems, there may be buttons to press or similar inputs that influence the data exchanged over the air.

Having software tools such as **Universal Radio Hacker (URH)** allows us to easily take quick captures, name them, and repeat the procedure for different conditions.

At this point, we've already known the frequency but not much else about the signal. We haven't been able to determine if the "Car keyfob" and "Smart light" RC uses frequency-shift keying (FSK), on-off keying (OOK), or another modulation scheme, but at this point, it's not yet crucial to find out. Before we can actually start capturing, we first need to guess the bandwidth and confirm the "exact" carrier frequency, of which we have a hint from the label. We use gr-fosphor to inspect the spectrum and the waterfall; the high resolution of gr-fosphor makes it a great reconnaissance tool. After finding the carrier frequency and bandwidth, we tune the BladeRF slightly toward the left to avoid any DC noise — a technique called offset tuning — and start to capture.

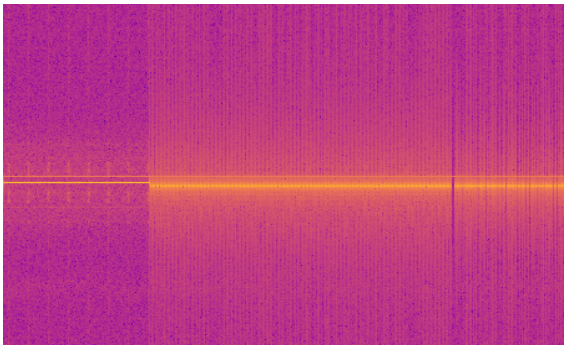


With the captured signals, it's possible to test for replay attack vulnerability. Moving away from the more powerful transmitter, we replay the different signals corresponding to specific effects to see if the car and light will respond. Sure enough, the light begins to flash on and off, and the door was unlocked.

RF Analysis

Having enough captures and a working receiver with us, we're able to continue our research back in our labs. Using the captured signals, we can then begin a more in-depth analysis. To do so, we need to clean or filter the signals to isolate only the signal that we need. This section largely involves the different features of URH that help filter the desired packets.

Filtering or noise-canceling



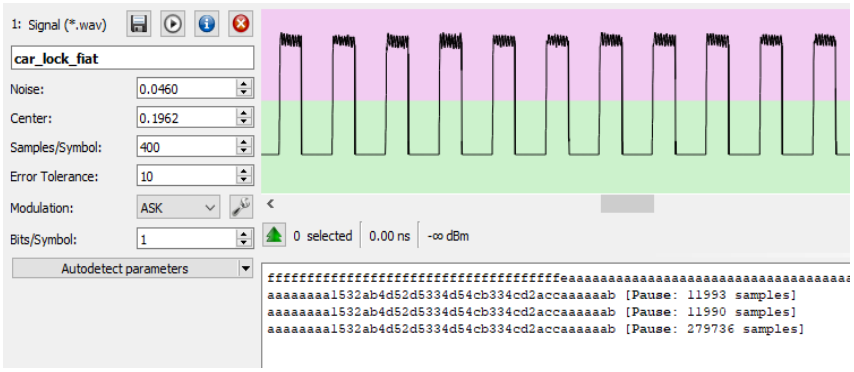
The signals we captured were not perfect, as we had no control over the RF-noisy environment. We had to purposely use offset tuning to avoid DC spikes. Thus, we need to re-center the signal.

URH also has a de-noising feature. From the analog view, we're able to find the "silence" zone. Selecting this zone and setting it as "noise" makes the red area around the signal get narrower or ideally just a bit larger than the "noise," as seen in Figure 8. This method does not create a copy of the signal, but it's still easy to copy and paste an entire signal in URH (memory permitting).

Demodulation

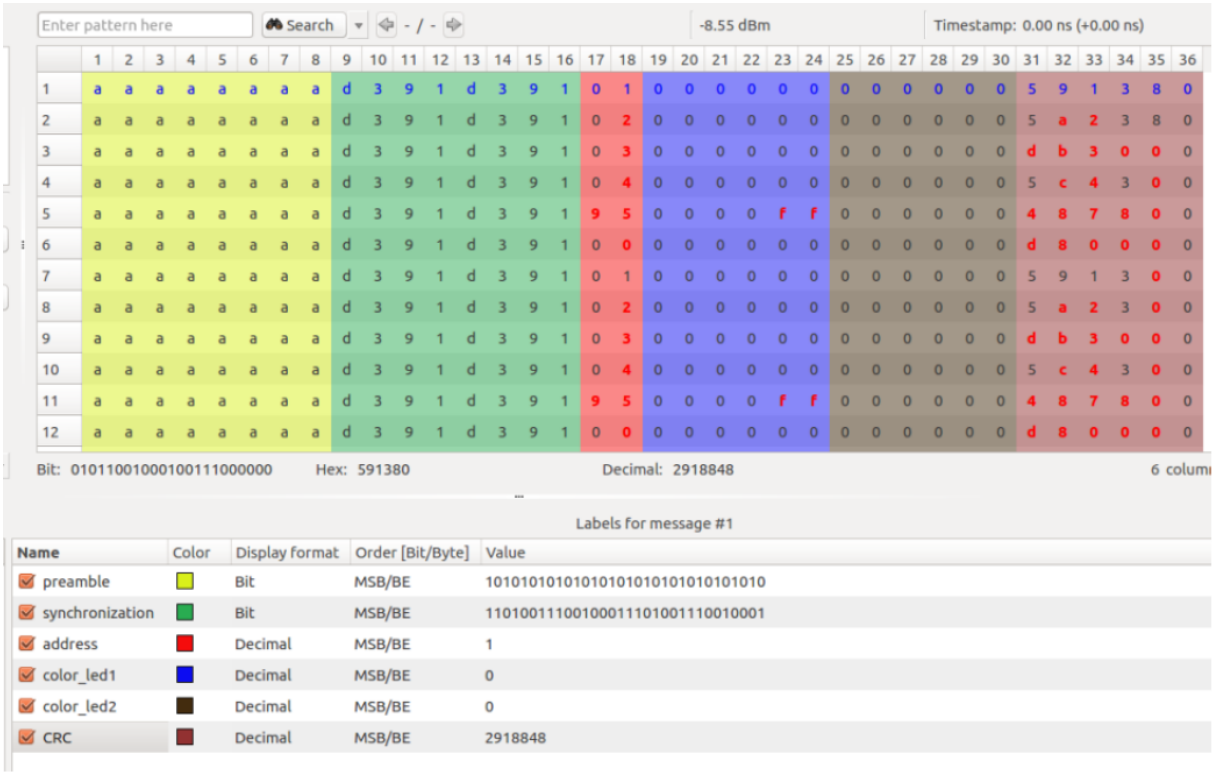
Since we're dealing with digital radio, we need to demodulate the analog signal to "see" the bits. In URH, we just switch to the "Demodulated" view, which results in the image in Figure below.

We don't apply a bandpass filter in this example so that the signal is still offset. Zooming in on this demodulated view reveals the packets in more detail, as seen in the bottom image of Figure 9. Since the center is still a bit offset, the demodulator interprets any sample of the signal above the threshold as 1. We just need to offset the threshold. A real (non-software) receiver does not require this adjustment since it's tuned to the exact carrier frequency and has built-in DC-noise cancellation subsystems, thereby making offset tuning unnecessary.



It's also possible to see something that looks like a preamble at the beginning of each packet. The preamble is necessary for any digital packet radio communication to "wake up" the receiver and provide a reference to set the symbol rate. In other words, the initial part of the packet is known and has to be "101010 …" (unless configured differently).

We use the preamble as a reference to set the symbol length (or bit length), assuming that one bit is one symbol in the alphabet. To do this, we select the shortest “pulse.” We find the shortest “symbol” in the preamble by choosing the point of the signal that crosses the threshold upward up to the point where the signal crosses back below the threshold.



Attack Classes

1. Replay Attack

   The attacker records RF packets and replays them to obtain basic control of the machine.

2. Command Injection

   Knowing the RF protocol, the attacker can arbitrarily and selectively modify RF packets to completely control the machine.

3. E-stop abuse

   The attacker can replay e-stop (emergency stop) commands indefinitely to cause a persistent denial-of-service (DoS) condition.

4. Malicious Repairing

   The attacker can clone a remote controller or its functionality to hijack a legitimate one.

Mitigation: Replay Attacks in Remote Keyless Entry Systems

1. Fixed predetermined authentication code
2. Rolling-code mechanism
3. Timestamp + XOR logic

Overview of the safety features and possible mitigations in remote controllers

Safety feature	Description	Issues prevented	Limitation
Pairing mechanism	Transmitter and receiver are paired with a (fixed) pairing code, which is used to recognize and accept commands only from known transmitters.	Interferences: Multiple transmitters (e.g. of the same model and brand) can work together in the same RF band.	Knowledge of the pairing code allows complete impersonation of a legitimate transmitter.
Passcode protection	The operator needs to enter a sequence (passcode) to operate the transmitter. This sequence enables the transmitter and starts the receiver.	Unwanted commands and unauthorized operators: Machinery can be controlled only upon entering the correct passcode.	Knowledge of the passcode allows anyone to use a transmitter.
Authorization	The transmitter implements an access control model that selectively enables or disables advanced features according to the level of the operator, who is identified using radio frequency identification (RFID) or an equivalent factor.	Inexperienced operators who might issue complex commands that could cause injuries.	RFID and equivalent factors can be stolen or cloned.
Virtual fencing	Transmitter and receiver communicate via an out-of-band channel (e.g., infrared) in addition to RF. When the transmitter is out of range, the receiver does not accept any commands.	Machines cannot be operated outside the "virtual fence" created by the out-of-band channel (e.g., the infrared range).	Knowledge of the out-of-band virtual fencing protocol allows mimicry of it.

Conclusion

- Patterns of Vulnerability identified:
    - No rolling-code
    - Weak or no encryption at all
- Need for security programs/awareness in the field of IIoT
- A security-enhanced RKE system using timestamping and XOR encoding are effective to defend replay and jamming-and-replay attacks.
- In addition, the XOR encoding system can protect the synchronized counter and verification code in situations where the AES key is compromised.

References

1. Syverson, Paul. (1994). A taxonomy of replay attacks [cryptographic protocols]. 187 - 191. 10.1109/CSFW.1994.315935.
2. Samy Kamkar, RollJam Attack https://www.wired.com/2015/08/hackers-tiny-device-unlocks-cars-opens-garages/
3. Greenberg, A. (2015, August 6). This Hacker's Tiny Device Unlocks Cars And Opens Garages. Retrieved December 15, 2015, from http://www.wired.com/2015/08/hackers-tiny-deviceunlockscars-opens-garages/
4. Krebs. (2015, October 14). Krebs on Security. Retrieved December 15, 2015, from http://krebsonsecurity.com/2014/10/replay-attacks-spoof-chip-card-charges/
5. Microsoft. (2015, December 2). Replay Attacks. Retrieved December 15, 2015, from https://msdn.microsoft.com/en-us/library/aa738652(v=vs.110).aspx
6. Telecare. (10 April 2018). Telecare. "Telecrane ASUS tablet with original software." Last accessed on 16 August 2018 at https://www.telecrane.it/en/tablet-with-original-software/.
7. Elgato Systems LLC. (26 March 2013). FCCID.io. "SHUN HU TECHNOLOGY CO., LTD FCC Wireless Applications." Last accessed on 16 August 2018 at https://fccid.io/RN4.
8. HBC-radiomatic GmbH. (17 December 2001). FCCID.io. "HBC-radiometric GmbH FCC Wireless Applications." Last accessed on 16 August 2018 at https://fccid.io/NO9.