Cyber Attacks on Smart Cars using SDR

Yash M. Kenia

B.Tech (Computer), Cyber Security Analyst, Mumbai, India

---***---

Abstract - There is a tremendous growth towards the concept of Smart Cities. As these highly efficient technologies flood the globe, there are certain security risks that arise in the due course. With high availability and efficiency arise new vulnerabilities and risks in the field of Information Security. This paper explores the concept of securing the Smart Keyless Cars which use radio frequencies to lock and unlock doors. This paper aims at the vulnerabilities of these keyless smart cars and hacking and exploiting them through the technologies like software defined radio (SDR) using software and devices like GNURadio and HackRF. Further, the countermeasures and mitigation models are also drafted for safeguarding the keyless cars.

Keywords - Smart Cities, Cyber Security, Information Security, Smart Devices, and Keyless smart cars.

I. INTRODUCTION

The traditional key method of locking and unlocking the car required manually inserting the key in the lock. This method was a bit tedious. To ease it, came a concept of keyless cars. This system did not require the user to manually enter the key in the lock to lock or unlock the door. Instead only a button had to be pressed on the key. It uses radio frequencies to lock and unlock the car. When the button on the key is pressed, a specific frequency is emitted by the keys. When the emitted frequency of the key matches with the car's required frequency, the doors are locked or unlocked.

A Software-defined radio (SDR) is a radio communication system which is implemented by software in an embedded system or a computer instead of using the modulators or amplifiers. A common software used for the implementation of SDR is GNURadio Companion. [2]

GNURadio is a software development toolkit which pro- vides signal processing blocks for the implementation of SDR and radio signalling and tuning. An external RF hardware can be attached to the computer and the GNURadio provides the interface to perform the logical implementation using these radio frequencies and signals. A commonly used RF hardware is HackRF. It provides a graphical user interface (UI) for the user to build flow graphs. [2]

HackRF is a hardware gadget developed by Great Scott Gadgets which works well with GNURadio for implementing the Radio Frequencies, replaying, tuning and signalling. It is a peripheral for the GNURadio companion. The whole concept of SDR can be implemented using GNU and HackRF.[2]

This paper shows the concepts of Software-defined radio on the keyless smart cars for capturing and replaying the key frequencies with the concept of Cyber Security attack like Man-in-the-middle attacks and replay attacks.

II. PROPOSED SYSTEM

A. Attack Models

1) Man in the middle attack (MITM)

In Information security, Man in the middle attack is a type of attack in which the attacker secretly relays or alters the communication between two victims who are made to believe that they are directly communicating with each other. The victims are completely unaware of the presence of attacker in the middle. MITM attacks can be either passive or active. Passive MITM attacks are done just to constantly sniff the traffic between two parties. Active MITM attacks are done for the purpose of harming or causing damage to the victims.

2) Replay attack

A replay attack in Information Security is a type of attack in which a data transmission between two parties is captured in between and is replayed with or without manipulating it without the knowledge of the two parties. This attack is similar to

MITM attack and is the lower tier version of it.

3) Relay attack

A relay attack is similar to man in the middle attack in which the communication is initiated by the attacker in which the attacker simply relays the message without even manipulating it.

B. Threat Models

Threat models are typically made during the product de- velopment and design process. If a company producing a particular product has a good development life cycle, it creates the threat model when product development begins and continuously updates the model as the product moves through the development life cycle. These models are living documents that change as the target changes and as you learn more about a target, so you should update them often.[3]

The high-level threats are that an attacker could:

- Remotely take over a vehicle
- Shut down a vehicle
- Spy on vehicle occupants
- Unlock a vehicle
- Steal a vehicle
- Track a vehicle
- Thwart safety systems
- Install malware on the vehicle

Key Fob An attacker could exploit the key fob connection to:

- Lock out a key
- Brute-force the key fob algorithm
- Clone the key fob
- Jam the key fob signal [3]

III. IMPLEMENTATION

This explains the implementation of the Hacking of the keyless smart cars using the technology of Software-defined radio – GNURadio and HackRF. Two phases have been explained here:

A. Implementation and Exploitation.

1) Capturing the radio frequencies of the key

The following is the flow graph which is used to capture the frequencies:

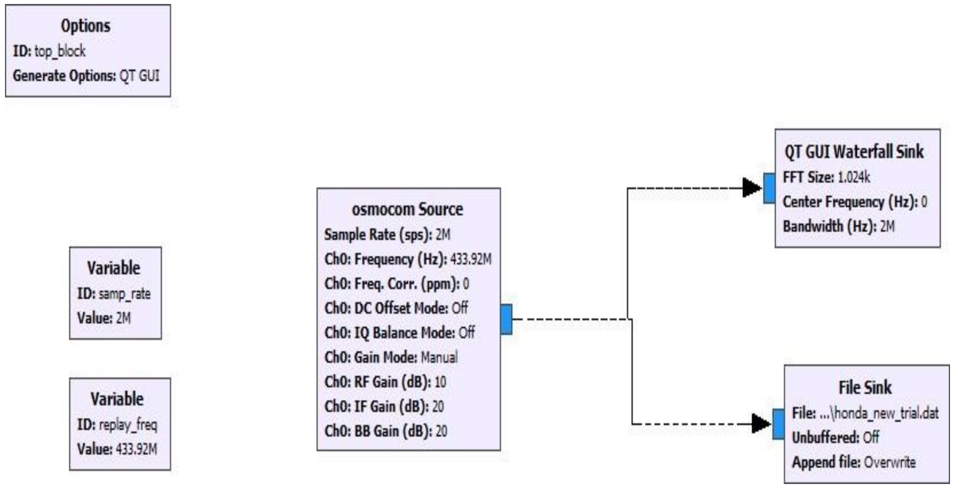
Options: ID is set to the top block and the Generate Options is set to QT GUI. Which is a type of frequency graph for the graphical user interface. These are the basic and default parameters which are set in the GNU Radio Companion.

Osmocom Source: The Osmocom source is an abstraction layer that allows us to communicate with different hardware devices (HackRF in our case) for software radio. And it is a source which produces digital signals that will be consumed by the next block in the flow graph. This tells the HackRF to switch to the receiving mode via the USB. It has different parameters like the sample rate which we have specified in the earlier stage. We also specify the channel frequency of approximate signal and the car which is set to 433.9M Hz. The RF Gains are set to 0 to avoid any errors and actually are not used in our demo.

QT GUI Waterfall Sink: This is the graphical user interface for the user which will show a graphical structure and show the details of the frequencies that are being emitted at every second by the rates of 2M Hz. We define different parameters like the Center Frequency which is set to 0 by default and the Bandwidth for the flow graph in 2M Hz which is kept as the same for the sample rate.

The captured frequencies are further stored in a file format in the computer using HackRF and GNU Radio Companion which are based on the technology of Software Defined Radio. [6]

The peak denotes the captured frequency of the car key.



Options
ID: top_block
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 2M

Variable
ID: replay_freq
Value: 433.92M

osmocom Source
Sample Rate (sps): 2M
Ch0: Frequency (Hz): 433.92M
Ch0: Freq. Corr. (ppm): 0
Ch0: DC Offset Mode: Off
Ch0: IQ Balance Mode: Off
Ch0: Gain Mode: Manual
Ch0: RF Gain (dB): 10
Ch0: IF Gain (dB): 20
Ch0: BB Gain (dB): 20

QT GUI Waterfall Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 2M

File Sink
File: ...\honda_new_trial.dat
Unbuffered: Off
Append file: Overwrite

Fig.1. Flow graph for captured frequency

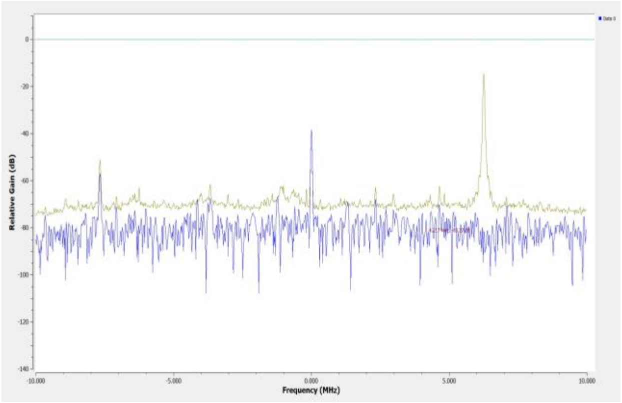
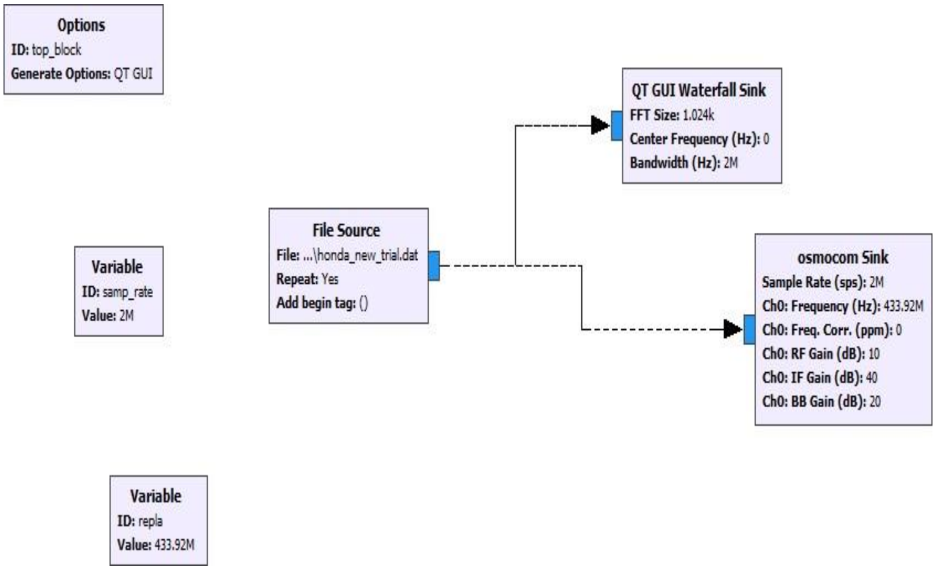


Fig. 2. Captured frequency

2) Replaying the captured frequencies directly on the car

Throttle Block: This is the block to refine the frequencies and emit them equally and repeatedly on the car i.e. to match the required frequency to crack the lock.



Options
ID: top_block
Generate Options: QT GUI

Variable
ID: samp_rate
Value: 2M

Variable
ID: repla
Value: 433.92M

File Source
File: ...\honda_new_trial.dat
Repeat: Yes
Add begin tag: ()

QT GUI Waterfall Sink
FFT Size: 1.024k
Center Frequency (Hz): 0
Bandwidth (Hz): 2M

osmocom Sink
Sample Rate (sps): 2M
Ch0: Frequency (Hz): 433.92M
Ch0: Freq. Corr. (ppm): 0
Ch0: RF Gain (dB): 10
Ch0: IF Gain (dB): 40
Ch0: BB Gain (dB): 20

Fig. 3. Replay flowgraph

These frequency flow graphs are replayed over the car repeatedly and the car unlocks.

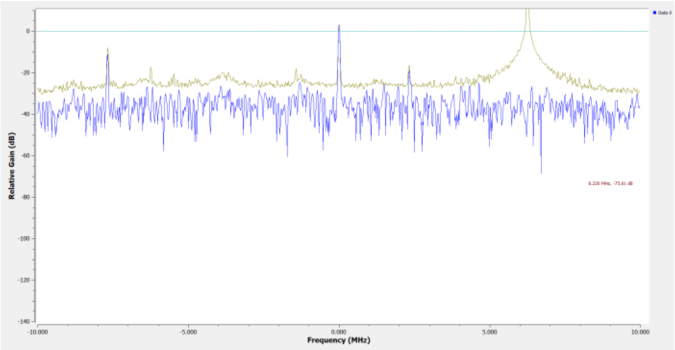


Fig. 4. Replayed frequency

B. Mitigation Models and Counter Measures

1) Rolling Code

A rolling code also known as hopping code is a mechanism used in keyless smart cars and systems to avoid the replay and relay attacks, where in an attacker captures the radio frequency transmissions and replays it to cause the system or the car to unlock. These applications are widely seen in normal car doors and keyless entry systems. [4]

A rolling code in keyless section frameworks to mitigate and avoid replay attacks, where a meddler records the transmission and replays it at a certain time to make the victim 'open' the lock and use the key to open the car and capture the transmitted frequency.

Techniques:

1. A pseudo random number is generated which is crypto- graphically secure from both the receiver and transmitter end.
2. Transmitter sends next number code in sequence.

3. Receiver compares this to its calculated next number code.

4. A typical implementation compares within the next 256 codes in case receiver missed some transmitted keys[4]

2) KeeLoq

KeeLoq"code hopping" encoders encrypt a 0-filled 32-bit block with KeeLoq cipher to produce a 32-bit "hopping code". A 32-bit initialization vector is linearly added (XORed) to the 32 least significant bits of the key prior to encryption and after decryption.

Figure 2 shows the structure diagram of KEELOQ Encryption. The Counter which is used to shield the frequency numeric code from being captured. Once this encoder detects the press of the button, it reads the input and increments the Sync Counter value. The counter and keys are input to the encryption algorithm and the output is a 32 bit data. This will change with every press of the button of the key fob. And its value will be randomly hoping and generated. Hence, this is the hopping portion of the data or the code. This 32 bit is combined with a fixed 34 bit portion which consists of the information about the serial number from which the code is transmitted to the receiver. [1]

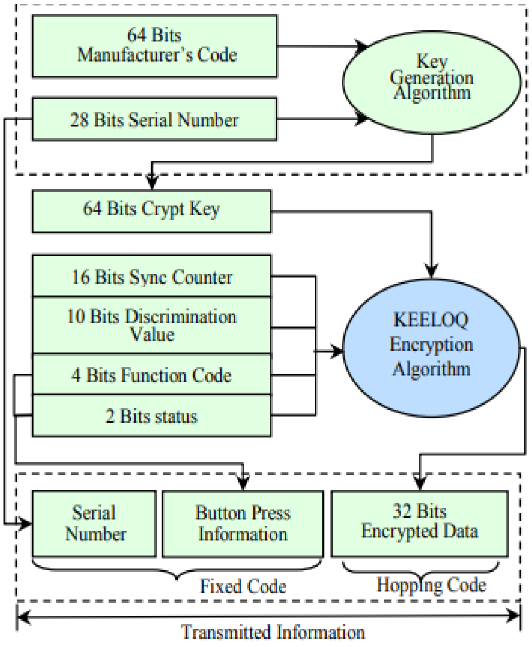


Fig. 5. KeeLoq Encryption Algorithm

IV. EXPERIMENTAL RESULTS

The experimental results are all part of the flow graphs in the GNURadio of the computer which was connected with the HackRF. The frequency of the key was captured and re- played using the above concepts of software-defined radio. The test subject was the Indian version of Honda City i-vtec. This research and experiment has also been performed on Toyota Innova Crysta and Maruti Wagon R. The experiment was successful in performing the man-in-the-middle and Replay attack.

V. CONCLUSION AND FUTURE WORK

This paper explains how the man in the middle attacks or replaying attacks are possible on the cars and vehicles using tools and techniques and frequency capturing by HackRF. Further, we also see that the same can be countered by the use of KeeLoq algorithm and rolling codes. This provides enough concepts and information on which more such vulnerabilities and loopholes should be patched and every car should at least have a rolling code mechanism in their locking and unlocking.

REFERENCES

1. Y. Hu, Y. Zhang and B. Sun,"Design of RKE System Based on KEELOQ Encryption Technology," 2009 International Conference on Artificial Intelligence and Computational Intelligence, Shanghai, 2009, pp. 324-327.

2. K. Hein äaro, "Cyber attacking tactical radio networks," 2015 International Conference on Military Communications and Information Systems (ICMCIS), Cracow, 2015, pp. 1-6. doi: 10.1109/ICMCIS.2015.7158684

3. The Car Hacker's Handbook: A Guide for Penetration Testers by Craig Smith. ISBN-10: 1-59327-703-2 ISBN-13: 978-1-59327-703-1 Publisher: William Pollock [4]https://en.wikipedia.org/wiki/Rolling code

4. Oka, Dennis Kengo, et al. "Survey of vehicle IoT bluetooth devices." Service- Oriented Computing and Applications (SOCA), 2014 IEEE 7th International Conference on. IEEE, 2014.

5. https://greatscottgadgets.com