Lab 2: BSM Spoofing Attack

Introduction

In V2V communication, safety-related applications are focused on the *basic safety message* (BSM). BSMs are broadcast by all V2V-enabled vehicles on a periodic basis (usually every $100ms$) and contain a variety of information about the sending vehicle's movements. While they can carry additional information as well, the core data contained in BSMs – the sender's GPS position, speed, and direction of travel – is essential for allowing other vehicles to be aware of the sender's movements and take any necessary action to avoid a potential collision. The situational awareness and corresponding collision avoidance capabilities provided through BSM exchange are among the most significant benefits of V2V technology.

Unfortunately, the use of BSMs also creates a new attack surface. Malicious attackers can pursue a variety of attacks against BSMs, and the potential consequences may vary from significantly disrupting traffic flow to actually causing (rather than avoiding) a collision. In this lab, you will explore one major attack against BSMs. In a *BSM spoofing* attack, the attacker transmits fake BSMs that are made to look as realistic as possible in the hopes that receiving vehicles will believe the spoofed messages are genuine and take unnecessary actions based on their contents. You will use the V2Verifier testbed to execute a BSM spoofing attack and explore not only how such attacks work, but also how V2V security mechanisms provided in the IEEE 1609.2 standard are used to detect and mitigate such attacks.

Required Equipment

As in Lab 1, you may choose to complete this lab using either DSRC or C-V2X. Ensure you have the necessary equipment for your chosen technology before proceeding with lab. Note that this lab requires 3 USRPs (and associated peripherals, e.g., GPS antennas) rather than the two required for Lab 1.

<table>
<tr><th>DSRC</th><th>C-V2X</th></tr>
<tr><td>3x USRP B210 SDRs

---</td><td>3x USRP B210 SDRs
3x TXCO GPSDO module (installed in B210s)
3x 3V GPS antennas (for B210s)
3x 5V DC power adapter for USRP</td></tr>
<tr><td colspan="2">3x antennas capable of operation on 5.9 GHz
3x PCs (or virtual machines) with USB 3.0 ports</td></tr>
<tr><td>---</td><td>1x LimeSDR</td></tr>
</table>

Operating System

This lab is designed to be used with Ubuntu 18.04. These instructions may not work properly with older or newer versions. An Ubuntu 18.04 ISO is available at http://releases.ubuntu.com/18.04/. Instructions for installing Ubuntu on a PC are available here. Although a PC is recommended, if using VMs you may refer to Appendix 1 for Ubuntu installation instructions.

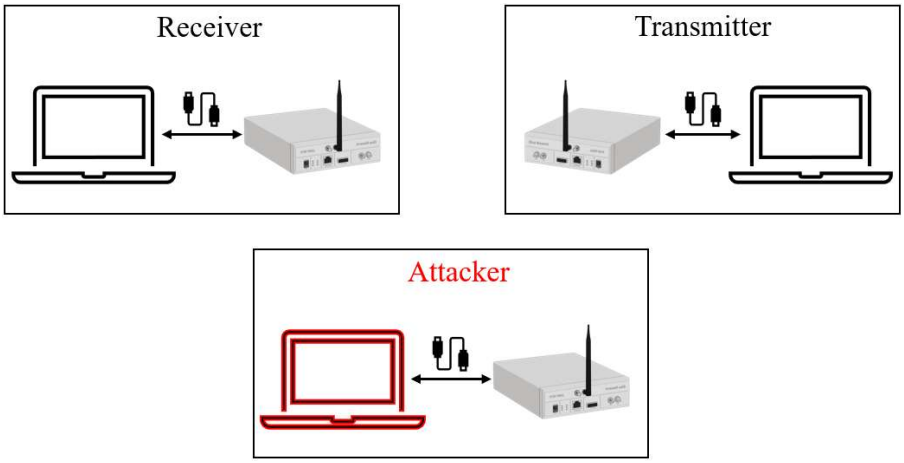
Activity 1 – Install the Operating System

Make sure you complete this activity on three PCs**.** If you are using VMs, you may choose to create one VM and then clone it for use on separate PCs (to save time).

1. Install Ubuntu 18.04 LTS. You can get an ISO from https://releases.ubuntu.com/18.04/.
    - If you are using a PC, follow the Ubuntu installation instructions.
    - If you are using a VM:
        i. Create a new **Ubuntu 18.04 LTS** virtual machine using VMware Workstation[1]. Do not attempt to download or use an existing Ubuntu VM, you **must** create a new VM for these instructions to work properly.
        ii. Give the VM a minimum of 4GB memory and 20GB storage and leave the remaining installation settings as their default values. Once installed, wait for the OS to automatically update, then reboot.
2. Run `sudo apt update` and `sudo apt -y upgrade` to make sure all installed software packages are up to date. Do not upgrade to a newer Ubuntu version if you are prompted to do so. When the update process completes, restart the PCs (or VMs) and proceed to the next activity.

Activity 2 – Installing the V2Verifier Testbed

This lab assumes that you have completed Lab 1 and either already have, or are already familiar with, the necessary steps to set up and configure the V2Verifier testbed for your chosen technology. If you have not completed Lab 1 (or just need a refresher), refer to Activity 2 in Lab 1 for detailed instructions. **Make sure to set up three (3) PCs** for this lab. You will need one PC each to transmit and receive BSMs, as well as a third PC to serve as the attacker. PCs will be referred to according to the diagram below as receiver, transmitter, and attacker in the remainder of this lab.



[1] If VMware Workstation is unavailable, we recommend using VirtualBox as a (free) alternative.

Activity 3 – BSM Spoofing Attack

This activity, like the previous one, is divided into two sections depending on your choice of DSRC (Activity 3a) or C-V2X (Activity 3b). Proceed accordingly.

Activity 3a (DSRC)

1. On the receiver and transmitter PCs, open a terminal and run `gnuradio-companion` to launch GNURadio.
2. On the receiver PC:
    - Open and run the Wi-Fi receiver flowgraph (`wifi_rx.grc`) that is in the `grc` directory of V2Verifier.
    - Open a new terminal and `cd` into the cloned V2Verifier directory.
    - Launch the V2Verifier receiver program with the option (`-g`) to launch its web-based graphical interface.

      ```
      python3 v2verifier.py receiver -t dsrc -g web -s false
      ```

3. On the transmitter PC:
    - Open and run the Wi-Fi transmitter flowgraph (`wifi_tx.grc`) that is in the `grc` directory of V2Verifier.
    - Open a new terminal and `cd` into the cloned V2Verifier directory.
    - Launch the V2Verifier transmitter program.

      ```
      python3 v2verifier.py transmitter -t dsrc
      ```

    - On the USRP, you should see a red blinking light on the port with the antenna attached (e.g., TX/RX) indicating that messages are being transmitted. If not, double check that your USRP is properly connected to the PC and that you have carefully followed all instructions above.

4. Returning to the receiver PC, you should see some activity on the V2Verifier GUI, confirming that BSMs are being sent and received. Make sure this is working properly before proceeding.
5. On the attacker PC:
    - Open the main V2Verifier execution script (`v2verifier.py`) in a text editor.
    - Find the line that invokes the `import_key` function. Change the provided value from its default ("keys/0/p256.key") to a different key (e.g., "keys/1/p256.key"). This will cause the attacker to attempt to use a key it does not actually own to sign messages – one form of a basic spoofing attack.
    - Save and close `v2verifier.py`.

- Open and run the `wifi_tx.grc` flowgraph
- In a new terminal, `cd` into the cloned V2Verifier directory.
- Launch a BSM spoofing attack with the following command:

```
python3 v2verifier.py transmitter -t dsrc
```

6. On the receiver PC, take a look at the V2Verifier GUI. What do you see that might be indicative of a BSM spoofing attack? Take note of your observations to include in your lab report.

Activity 3b (C-V2X)

Remember to verify that your three USRPs have GPS lock before continuing.

1. On the transmitter and receiver PCs, open a terminal and navigate to the V2Verifier C-V2X directory

```
cd ~/v2verifier/cv2x
```

2. On the receiver PC:
   - Launch the C-V2X receiver program `pssch_ue` with the command

```
./build/pssch_ue -a clock=gpsdo
```

   - Open a separate terminal window and launch the V2Verifier receiver

```
cd ~/v2verifier
python3 ./v2verifier.py receiver -t cv2x -g web -s false
```

3. On the transmitter PC:
   - Launch the C-V2X transmitter program `cv2x-traffic-generator` with the command

```
./build/cv2x_traffic_generator -a clock=gpsdo
```

   - Open a separate terminal window and launch the V2Verifier transmitter

```
cd ~/v2verifier
python3 ./v2verifier.py -t cv2x transmitter
```

4. Returning to the receiver PC, you should see some activity on the V2Verifier GUI, confirming that BSMs are being sent and received. Make sure this is working properly before proceeding.

5. On the attacker PC:
    - Open the main V2Verifier execution script (`v2verifier.py`) in a text editor.
    - Find the line that invokes the `import_key` function. Change the provided value from its default ("keys/0/p256.key") to a different key (e.g., "keys/1/p256.key"). This will cause the attacker to attempt to use a key it does not actually own to sign messages – one form of a basic spoofing attack.
    - Save and close `v2verifier.py`.
    - Launch the C-V2X transmitter program `cv2x-traffic-generator` with the command

      ```
      ./build/cv2x_traffic_generator -a clock=gpsdo
      ```

    - In a new terminal, `cd` into the cloned V2Verifier directory.
    - Launch a BSM spoofing attack with the following command:

      ```
      python3 v2verifier.py transmitter -t cv2x
      ```

6. On the receiver PC, take a look at the V2Verifier GUI. What do you see that might be indicative of a BSM spoofing attack? Take note of your observations to include in your lab report.

Activity 4 – Mitigating BSM Spoofing with 1609.2 Security

For this activity, you will repeat the previous activity, but this time you will enable 1609.2-based V2V security. Depending on your choice of technology, repeat steps 1-5 of Activity 3a or Activity 3b as given above; however, this time when you run the V2Verifier receiver, provide `true` as the argument for the `-s` option. Once you have executed a BSM spoofing attack with 1609.2 security enabled (`-s true`), answer the following questions:

1. On the receiver PC, look at the V2Verifier GUI. Do you see anything different this time that might indicate the (in)effectiveness of 1609.2 for detecting and/or mitigating the attack? What is different than it was before?
2. Based on your observations, do you think IEEE 1609.2 provides sufficient mitigation for BSM spoofing attacks?
3. On the receiving PC, open up the packet capture file in `/tmp/out.pcap` (DSRC) or `/tmp/ue.pcap` (C-V2X). Take a look in the packet contents (specifically, at the 1609.2 security fields). What field(s) do you see that are applicable to mitigating spoofing attacks? To answer this question, consider what information was provided on the GUI in this activity (with security enabled) as well as your understanding of how a spoofing attack works.