Lab 1: Introduction to V2V & V2V Security Testbed

Introduction

Vehicle-to-vehicle (V2V) communication has historically been difficult to evaluate in an experimental setting. Commercial V2V equipment is often expensive, and many manufacturers will not even sell to researchers (they prefer bulk sales directly to auto companies). Fortunately, thanks to the advent of software-defined radios (SDRs), we no longer need "real" equipment in order to experiment with V2V protocols.

In this lab, you will get your first hands-on experience with V2V communications through the *V2Verifier* testbed. V2Verifier, created in RIT's WISP lab, is an open-source SDR testbed for V2V security that allows experimental work with multiple V2V protocols in settings ranging from laboratories to in-vehicle experiments. The core objective of this lab is to learn how to set up the V2Verifier testbed, as well as to get familiar with some of the features of V2Verifier. Completing these objectives will prepare you to complete future labs in this course.

You may choose whether to complete this lab using Dedicated Short-Range Communication (DSRC) or Cellular Vehicle-to-Everything (C-V2X) as V2Verifier supports both protocols. When making your decision, note that C-V2X requires more equipment and is somewhat more complicated to configure. Ensure that you have access to the necessary equipment (see below) as well as sufficient time to complete the lab for whichever protocol you choose.

Required Equipment

DSRC	**C-V2X**
2x USRP B210 SDRs --- ---	2x USRP B210 SDRs 2x TXCO GPSDO module (installed in B210s) 2x 3V GPS antennas (for B210s) 2x 5V DC power adapter for USRP
2x antennas capable of operation on 5.9 GHz 2x PCs (or virtual machines) with USB 3.0 ports	
---	1x LimeSDR

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 – Installing the Operating System

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

Make sure to complete this activity on both PCs. Again, if you are using a VM, you may choose to complete these instructions on one VM and then clone it.

1. **Software dependencies**: Use the following command to install necessary software packages, include GNURadio, UHD drivers for working with the USRPs, and Python packages used by V2Verifier. Pay close attention to your commands, being particularly careful to use **pip3** rather than the obsolete **pip** command[2].

```
sudo apt install -y git cmake libuhd-dev uhd-host swig libgmp3-dev
python3-pip python3-tk python3-pil python3-pil.imagetk gnuradio

pip3 install -U fastecdsa pyyaml eel folium pynmea2
```

The remainder of this activity depends on your choice of technology. If you want to complete this lab using DSRC, proceed with Activity 2a. For C-V2X, proceed to Activity 2b.

[1] If VMware Workstation is unavailable, we recommend using VirtualBox as a (free) alternative.
[2] V2Verifier is under active development, so the prerequisite packages may change periodically. Be sure to check the installation instructions on the project README page, as updates there may not be immediately reflected in these instructions.

Activity 2a (DSRC)

1. To support IEEE 802.11p communication with the USRPs, you need to install two GNURadio modules from the open-source WiME project.

```
cd ~
git clone https://github.com/bastibl/gr-foo.git
cd gr-foo
git checkout maint-3.7
mkdir build
cd build
cmake ..
make
sudo make install
sudo ldconfig

cd ~
git clone git://github.com/bastibl/gr-ieee802-11.git
cd gr-ieee802-11
git checkout maint-3.7
mkdir build
cd build
cmake ..
make
sudo make install
sudo ldconfig
```

2. Clone the V2Verifier GitHub repository into your home directory.

```
cd ~
git clone https://github.com/twardokus/v2verifier.git
```

3. Open a new terminal and launch GNURadio Companion, the graphical interface for GNURadio.

```
gnuradio-companion
```

4. In GNURadio Companion, open the `wifi_phy_hier.grc` file in `~/gr-ieee802-11`. Click the green "play" button at the top of the window to execute this file.

5. Now, open the `wifi_loopback.grc` file from the same directory. Run this file. If execution proceeds without errors, you have completed configuration of V2Verifier for use with DSRC. Stop the flowgraph (click the red "stop" button at top) and proceed to the next activity. If you encounter any errors, you may consult the WiME project troubleshooting guide for assistance.

Activity 2b (C-V2X)

1. First, install some packages for LimeSDR. You will use a LimeSDR device to transmit a synthesized GPS signal that will allow the USRPs to synchronize in time.

   ```
   sudo apt install -y liblimesuite-dev liblimesuite17.2.1 limesuite
   limesuite-udev
   ```

2. Confirm that your PC can communicate with the LimeSDR by connecting the LimeSDR device – be sure to connect it to a USB 3 port – and running the command:

   ```
   LimeUtil --find
   ```

   You should see output indicating the connected device's serial number (and other information). If not, double check your connection. You may (unusually) need to restart the PC.

3. Now, download and install the GPS-SDR-SIM project. GPS-SDR-SIM is an open-source tool for generating synthesized GPS signals and includes an application to transmit such signals over the air using a LimeSDR device.

   ```
   cd ~
   git clone https://github.com/osqzss/gps-sdr-sim.git
   cd gps-sdr-sim
   gcc gpssim.c -lm -O3 -o gps-sdr-sim
   cd player
   make limeplayer
   ```

4. In order to synthesize a GPS signal, GPS-SDR-SIM requires a GPS ephemeris file[3]. You can obtain an ephemeris file from NASA's EarthData portal at https://cddis.nasa.gov/archive/gnss/data/daily/. Note that you may be required to register for a free account in order to access the site. Do so, if necessary, then navigate through the FTP portal to the directory for the current date and download the ephemeris file (*.brdc) for today's date.

5. Place the downloaded file in `~/gps-sdr-sim`, then run the following commands to synthesize a GPS signal and write it to a binary (.bin) file.

   ```
   cd ~/gps-sdr-sim
   ./gps-sdr-sim -e <brdc_file_name> -l 43.0766224,-77.6715616,100 -d
   7200 -o gpssim.bin -s 1000000 -b 1
   ```

[3] A more detailed description of how this works is available on the GPS-SDR-SIM README: https://github.com/osqzss/gps-sdr-sim/blob/master/README.md.

6. This process may take a few minutes to complete. While the signal file is being generated, take a look at the options (`-l, -d, -o, -s, -b`) for the previous command. Can you make an educated guess as to what each of these arguments provides?

7. Copy the output file (`gpssim.bin`) into the `player` subdirectory of `~/gps-sdr-sim`.

   ```
   cp gpssim.bin ./player
   ```

8. At this point, you are ready to begin transmitting synthesized GPS signals. Before doing so, you need to install and build the V2Verifier C-V2X code that will take advantage of these signals for time synchronization. Open a new terminal and run the following commands.

   ```
   cd ~
   git clone https://github.com/twardokus/v2verifier.git
   cd v2verifier/cv2x
   mkdir build
   cd build
   cmake ../
   make
   sudo make install
   ```

   If you are using a VM, stop here and clone it. The remainder of the activity requires two PCs.

9. Connect one USRP B210 with GPSDO to each PC via USB 3. *Make sure that <u>each</u> USRP has a GPSDO module installed and is equipped with a GPS antenna.* To ensure the GPSDOs are supplied with sufficient power, use the 5V power adapters to plug each USRP into a wall outlet. On each PC, run the command `uhd_find_devices` to ensure the PC can communicate with the USRP that is connected to it.

10. On the first PC, connect the LimeSDR via USB 3. Return to the terminal where you built GPS-SDR-SIM. Navigate to the `player` subdirectory, then run the `limeplayer` command to begin transmitting the synthesized GPS signal.

    ```
    cd ~/gps-sdr-sim/player
    ./limeplayer -s 1000000 -b 1 -d 2047 -g 0.1 < gpssim.bin
    ```

11. Wait a few minutes for the USRPs to obtain GPS lock. When this occurs, the indicator light on each USRP (next to the GPS antenna port) will turn solid green. Note that it may take up to ten minutes for the USRPs to lock on to the signal, so be patient!

12. Once the USRPs have GPS lock, you are prepared to use V2Verifier with C-V2X. Proceed to the next activity.

Activity 3 – Setting up a Basic Experiment in V2Verifier

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 each PC, open a terminal and run `gnuradio-companion` to launch GNURadio.
2. On one PC (the "receiver"):
    - Open the Wi-Fi receiver flowgraph (`wifi_rx.grc`) that is in the `grc` directory of V2Verifier.
    - If you are unfamiliar with GNURadio, take a look at the interface. GNURadio flowgraphs (*.grc files) consist of connected *blocks*, which are written in either Python or C++. In the `wifi_rx.grc` flowgraph, some of the options are specific to IEEE 802.11p. For example, the frequency (5.89 GHz) and channel bandwidth (10 MHz) are both uniquely set for V2V communication.
    - Run the `wifi_rx.grc` flowgraph.
    - 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
      ```

3. On the other PC (the "transmitter"):
    - Open the Wi-Fi transmitter flowgraph (`wifi_tx.grc`) that is in the `grc` directory of V2Verifier.
    - Run the `wifi_tx.grc` flowgraph.
    - 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 first (receiver) PC, you should see some activity on the V2Verifier GUI.
    - You can find several reports from the perspective of a vehicle that is receiving messages. What are the security parameters reported on the GUI? Include this information in your report.

5. Stop the GUI program with Ctrl+C in the terminal where you launched it. You may need to Ctrl+C a few times to fully stop the program. You may also stop the GNURadio flowgraphs on both PCs.
6. The GNURadio flowgraph creates a capture file in `/tmp/out.pcap` that contains all of the packets it received while running. Open this capture file in Wireshark.
7. Within the capture file, selecting any packet will display data for each protocol layer. You should see the V2V networking layers as well as the elements of the upper layer protocol. Take particular note of the security elements therein! (Hint: look for "IEEE 1609.2" in the packet). In your report, include this information along with a brief explanation of what each security element is and why it is important for V2V.

Activity 3b (C-V2X)

1. On each PC, open a terminal and navigate to the V2Verifier C-V2X directory

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

2. On the first PC (the "receiver"):
   - 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
```

3. On the second PC (the "transmitter"):
   - 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

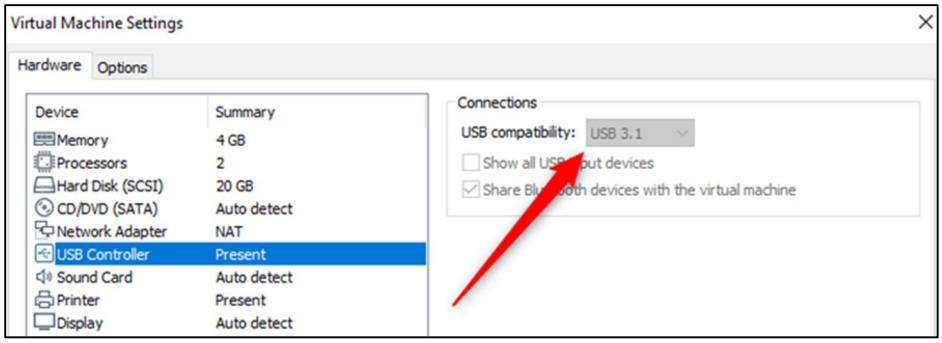
```
cd ~/v2verifier
python3 ./v2verifier.py -t cv2x transmitter
```

4. Returning to the first (receiver) PC, you should see some activity on the V2Verifier GUI.
   - You can find several reports from the perspective of a vehicle that is receiving messages. What are the security parameters reported on the GUI? Include this information in your report.

5. Stop the GUI program with Ctrl+C in the terminal where you launched it. You may need to Ctrl+C a few times to fully stop the program. You may stop the C-V2X programs in both terminals as well with Ctrl+C.
6. The C-V2X receiver creates a capture file in `/tmp/ue.pcap` on the "receiver" PC that contains all of the packets it received while running. Open this capture file in Wireshark.
7. Within the capture file, selecting any packet will display data for each protocol layer. You should see the V2V networking layers as well as the elements of the upper layer protocol. Take particular note of the security elements therein! (Hint: look for "IEEE 1609.2" in the packet). In your report, include this information along with a brief explanation of what each security element is and why it is important for V2V.

Appendix 1 – Installing Ubuntu 18.04 in a Virtual Machine using VMWare Workstation

1. Open VMware Workstation and create a new virtual machine using the ISO file you downloaded from the Ubuntu repository. The default memory, hard drive, and other resource allocations will be sufficient for this lab.
2. Boot up the VM and follow the prompts to install Ubuntu 18.04 just like you would on a physical computer.
3. Once the installation is complete and you have rebooted the VM, open the settings menu and make sure the USB controller is set to be compatible with USB 3.1. USB 3 is required to provide power and sufficient data rates for connecting the USRPs to a VM.



Appendix 2 – Configuring USB access for USRP B210s

1. First, you need to give non-root users access to USB devices. Make sure the USRP is **not** connected to the PC, then run the following commands.

```
sudo cp ~/gnuradio38/lib/uhd/utils/uhd-usrp.rules /etc/udev/rules.d/
sudo udevadm control --reload-rule
sudo udevadm trigger
```

2. Connect the USRP to the PC using a USB 3 port. If you are using a VM, forward the USB connection from the host to the virtual machine.

3. Download the latest USRP firmware images.

```
uhd_images_downloader
```

4. Run the command `uhd_find_devices`, then the command `uhd_usrp_probe`. These commands should show the device is a B210 USRP, and the latter will give you technical details about the device configuration. Peruse this information in as much depth as you wish; for this activity, the important part is that both commands detect the device and do not give any error messages.