TransFi user guide

Prerequisite:

1. Atheros_CSI_tool: https://github.com/xieyaxiongfly/Atheros_CSI_tool_OpenWRT_src/wiki. This is a custom driver that allows direct packet load injection
2. Lorcon* https://github.com/kismetwireless/lorcon. This program is used to inject the payload into the Atheros wireless card.
3. Ubuntu 14.04 (Other versions will not work)
4. Matlab any version > R2017b
5. AR9580: We used NETLEY Wireless N 5GHz 450Mbps 3*3MIMO WIFI CARD. https://us.amazon.com/NETELY-Wireless-Stations-PCIE-Card-Qualcomm-NET-N450B/dp/B07D8MRXRZ (Have not tested other models, but there could be some hardware differences between different manufactures.)
6. WARP v3 Boards: http://mangocomm.com/products/kits/warp-v3-kit/ with FMC-RF-2X245: Dual-Radio FMC Module. http://mangocomm.com/products/modules/fmc-rf-2x245/
7. Xilinx Design tool 14.4 (Newer version will not support WARP)

*Note: Lorcon is an old tool, there are compatibility issues. So please make sure it works after installation.

User guide

System requirement:

Any PC that supports PCIE Wireless adapters. (Not have any specific requirement of RAM or CPU.).

OS: Ubuntu 14.04 with ATH9K driver. No other modifications needed for the OS.

Preparation:

1. (Optional)Modify the IEEE80211_MAX_FRAME_LEN & IEEE80211_MAX_DATA_LEN in ieee80211.h within Atheros_CSI_tool: https://github.com/xieyaxiongfly/Atheros-CSI-Tool/include/linux/ieee80211.h (Currently support up to 7000 Bytes)
2. Then Install the Atheros_CSI_tool following guide from here: https://github.com/xieyaxiongfly/Atheros_CSI_tool_OpenWRT_src/wiki
3. Install Lorcon via : https://github.com/kismetwireless/lorcon, follow the guide here http://blog.opensecurityresearch.com/2012/09/getting-started-with-lorcon.html to install.
4. Make sure all the installation are correct. Check the custom kernel version number.
5. Check the WiFi interface ID via ifconfig. Record the interfance ID for further wireless card configuration
6. Use Emulation/Injection/Monitor_Channel_Set in command line to configure the Atheros chip operating in monitor mode and setting the wireless channel to operate on. The operating channel is set at line 13 with channel index 40 (5200MHz), 40Mhz channel is set with HT40-. For example, in current file, the channel is set with 40 HT40-, which is the 40MHz channel at channel 38 in this list: https://en.wikipedia.org/wiki/List_of_WLAN_channels. To specific the device to change the channel, replace wls4/wlp8s0 with your WiFi interface ID in line 13.
7. Set scrambler seed: Set npackets =1000 at line 46 in Emulation/Injection/scrambler_ini.c. Compile and run Emulation/Injection/scrambler_ini, and at the same time, run Emulation/Scrambler_calibration.m. Scrambler_calibration.m outputs the number of additional frames need to be transmitted to set the seed to 1 with difference variable at line 255.
   Change the npackets = difference in scrambler_ini.c. Recompile and run scrambler_ini to set the next scrambler seed at 1. (This should be done **ONLY ONCE** per restart of the PC.)

Note: I have prepared the predefined payload for scrambler initialization in Emulation/Injection/scrambler folder. You can inject those files for initialization with any emulation.

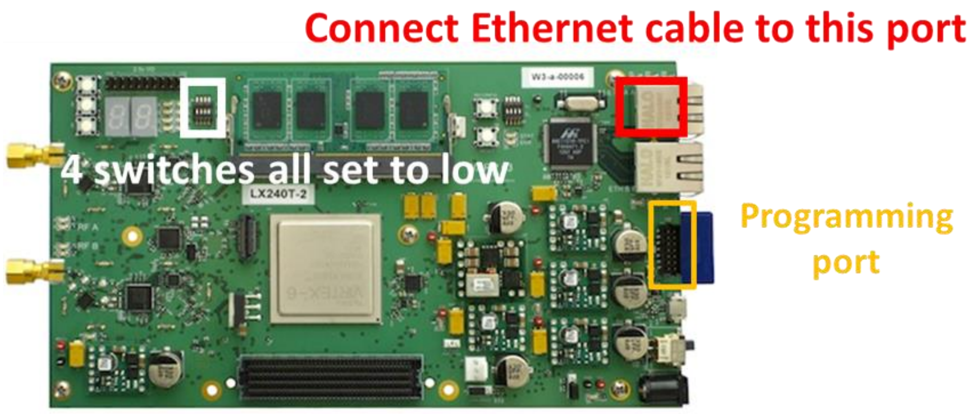
Use TransFi:

1. Go to Emulation/Find_packet.m. Generate the target signal you want before line 280. And give the target signal to the target_signal variable at line 280.
2. Then run Emulation/Find_packet.m, it will compute the MAC payload and output to the injection program. The output MAC payload is in MAC_PayloadXXX.txt. The XXX is number from 001 to 127.
3. Inject the MAC payload via running Emulation/Injection/injection_40M. You can configure the data rate with mcs_iter at line 180(2Tx MIMO 64QAM with 5/6 coderate is 15. Full table is here: https://en.wikipedia.org/wiki/IEEE_802.11n-2009.). The total number of transmitted frames is set at ntimes at line 8, which is ntimes*127.
4. Configure the device you want to inject in via line 43 in Emulation/Injection/injection_40M with *interface = “Your Interface ID”.
5. Compile the program with gcc -o Injection_40M Injection_40M.c -lorcon2.
6. Run the script with sudo ./ Injection_40M -i **** (**** = WiFi card interface name) . (Please use sudo)

Receiver configuration:

1. WARP configuration:

   A. If not, Xilinx Design tool 14.4 is need. (Newer Version does not support this FPGA core.)
   B. Then download this reference design from WARP http://warpproject.org/dl/refdes/warplab/v7/release/WARPLab_Reference_Design_7.7.1.zip
   C. Use the iMPACT software to program the board with w3_WARPLab_7.7.1_2RF.bin. To program the board with the programming port as shown here:


Connect Ethernet cable to this port
4 switches all set to low
Programming port

E. Once this is done follow the steps in 2.

2. WARP hardware configuration

A. The WARP boards are not configured correctly. First, a 1Gb internet switch is need. Set the IP address for you NIC at 10.0.0.250. Connect switch to the NIC and Warp to the switch in the ETH_A port as shown in previous picture.
B. Switch the DIP switch on WARP board to 0, the location of the switches are shown in previous picture. (All the 4 switch are set to low.)
C. Here is general user guide from WARP website: http://warpproject.org/trac/wiki/WARPLab/QuickStart

3. Setup MEX connection:

A. Within MATLAB, change your directory to the M_Code_Reference directory that you unpacked as part of the WARPLab release.
B. Run `wl_setup` in order to set up your MATLAB paths. When running `wl_setup` , you should only see the java transport available. If not, please check your paths to make sure there are no issues.Now your paths should be set up to proceed. Run `which wl_mex_udp_transport` to check that you can see the mex files. You should see: `<your WARPLab install dir>/M_Code_Reference/mex/wl_mex_udp_transport.m`
C. Run `wl_mex_udp_transport` from the MATLAB command line.
D. Follow this guide: http://warpproject.org/trac/wiki/WARPLab/MEX

4. If all the steps above are followed, the board should be functioning. If the board still cannot be connected, try multiple times or the board may be damaged during transport.

Using WARP as receiver

1. Configure the receiver with the corresponding target signal. Use WARP v3 boards as the receiver. You can write your own custom receiver program based on your target signal designs. The program used in the paper is in: Emulation/MISO_CustomReceiver.m.

2. The Emulation/MISO_CustomReceiver.m configures the WARP to detect the emulated signal with custom signal. The program will detect the emulated signals transmitted from the commodity devices. Configure your custom signal before line 517 and give your custom signal to custom_preamble_time variable. The detection threshold is set at LTS_CORR_THRESH. The detection results are shown in plotted figures. The general result is computed by counting the number of emulated frames that is being detected among all the figures. This program will also output SNR and bit sequence from the emulated signal.

3. Decoding with more antennas: Use Emulation/Emulation_4Tx_1Rx.m and Emulation/Emulation_3Tx_1Rx.m. It is a full emulation with fully operational Tx/Rx. The scripts require more than 4 WARP devices to operate. The BER and SNR will be the output of these .m files.

Other files in the folder:

1. Emulation\wlanNonHTData_local.m is used to generate a target signal with 802.11g configuration.
2. Emulation\weight_bit.m is a subfunction used to generate the segmented trellis diagram.
3. Emulation\depuncture.m is a subfunction to remove X' bits.