Primary User Emulation (PUE) Attack in Cognitive Radio Networks

Software Recommended: NetSim Standard v13.2, Visual Studio 2022

Project Download Link:
https://github.com/NetSim-TETCOS/PUE_Attack_v13.2/archive/refs/heads/main.zip

Follow the instructions specified in the following link to download and setup the Project in NetSim:

https://support.tetcos.com/en/support/solutions/articles/14000128666-downloading-and-setting-up-netsim-file-exchange-projects

Introduction

Cognitive Radio (CR) is a promising technology that can alleviate the spectrum shortage problem by enabling unlicensed users equipped with CRs to coexist with incumbent users in licensed spectrum bands while causing no interference to incumbent communications. Spectrum sensing is one of the essential mechanisms of CRs and its operational aspects are being investigated actively. In a hostile environment, an attacker may modify the air interface of a CR to mimic a primary user signal's characteristic, thereby causing legitimate secondary users to erroneously identify the attacker as a primary user. We coin the term *primary user emulation (PUE) attack* to refer to this attack. There is a realistic possibility of PUE attacks since CRs are highly reconfigurable due to their software-based air interface.
We create a PUE attack by adding two incumbents in the scenario in NetSim. One of the incumbents represents a "real" primary user while the second represents a "Malicious" primary user.
Our next goal is to detect the PUEA by the secondary users. For example, purposes we have set the detection time as proportional to the distance of the secondary users from the malicious primary user.
The code given below is for an example implementation of PUE Attack.

Example

1. The **PUE_Attack_Workspace** comes with a sample network configuration that are already saved. To open this example, go to Your work in the Home screen of NetSim and click on the **PUE_Attack_Example** from the list of experiments.
2. The network scenario loads as shown below:

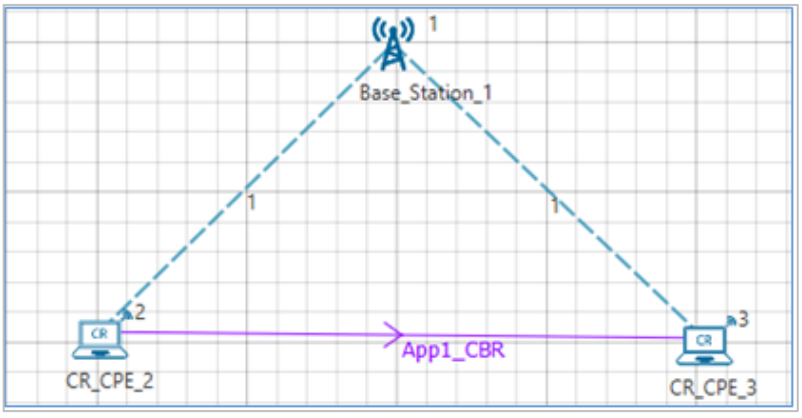


Figure 1: Network Topology

Settings done in example config file

3. Set the following property for **CR-Base_Station_1 → INTERFACE_1 (COGNITIVE_RADIO)** as shown in below given table.

Datalink Properties	
Incumbent count	2
Incumbent1(malicious) ON_Duration(s) OFF_Duration(s) Keep Distance(m)	 4 10 500
Incumbent2 (Incumbent) ON_Duration(s) OFF_Duration(s) Keep Distance(m)	 9 9 500
Physical Properties	
IFQP_Bitmap	1000000000000000

Table 1: CR Base Station 1 Properties

4. Distance between the CPE and Incumbent is < 500. This ensures that the incumbent is detected. If the incumbent is beyond the keep out distance, then it is not detected.

The timing diagram is as follows:

Malicious --- 0s to 10s (OFF), 10s to 14s (ON), 14s to 24s (OFF), 24s to 28s (ON) ... and so on
Incumbent --- 0s to 9 s (OFF), 9s to 18s (ON), 18s to 27s (OFF), 27s to 36s (ON) ... and so on

5. Now run the simulation 50 Sec.

Results and discussion

You can see the delay in the **CR_Detect.csv** file from the log files in the result dashboard Window. This additional delay has been set by the following code,

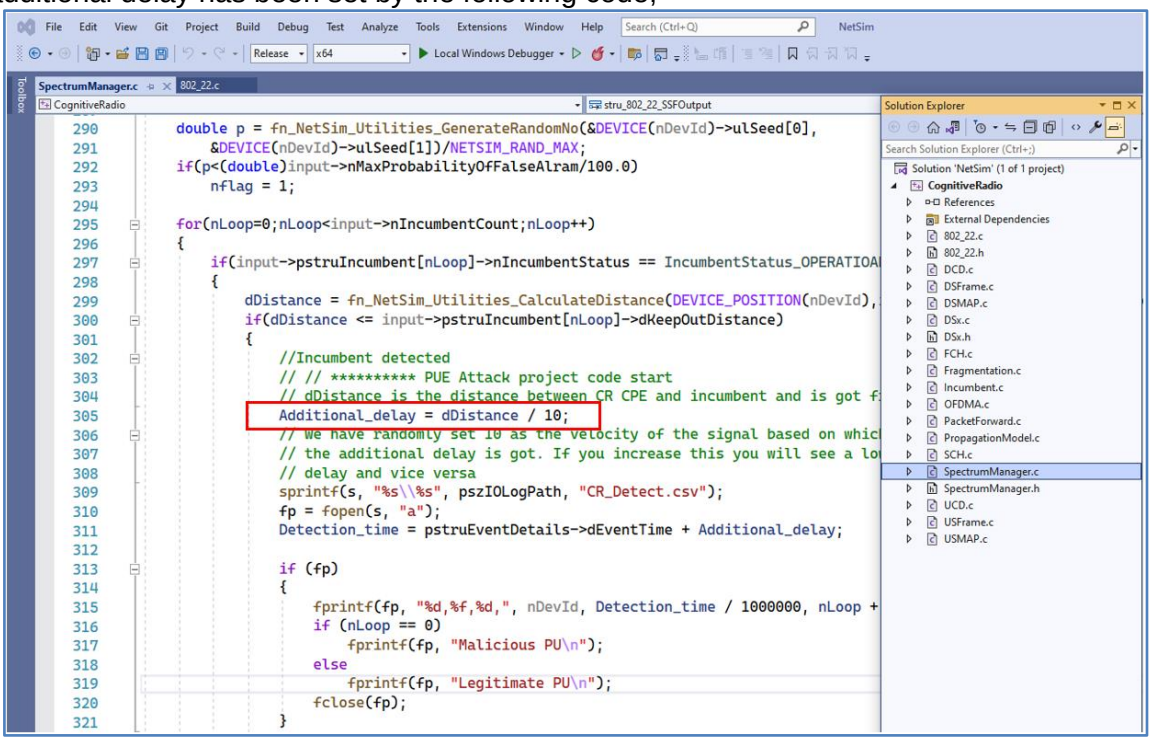
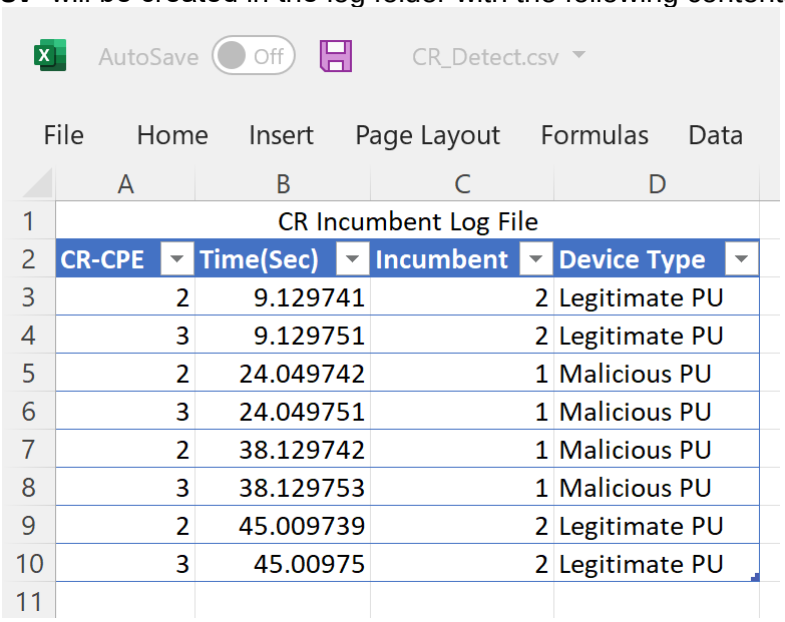


Figure 2 NetSim Project Source Code

User modifications can be made in the **fn_NetSim_CR_CPE_SSF()** function of **SpectrumManager.c** as mentioned below.
Additional_delay = dDistance / 10;
(You can also change the values as 10/100/1000 and analyses different variation in delay.)

A file "**CR_Detect.csv**" will be created in the log folder with the following contents:



	A	B	C	D
1	CR Incumbent Log File			
2	CR-CPE	Time(Sec)	Incumbent	Device Type
3	2	9.129741	2	Legitimate PU
4	3	9.129751	2	Legitimate PU
5	2	24.049742	1	Malicious PU
6	3	24.049751	1	Malicious PU
7	2	38.129742	1	Malicious PU
8	3	38.129753	1	Malicious PU
9	2	45.009739	2	Legitimate PU
10	3	45.00975	2	Legitimate PU
11				

Figure 3: CR_Detect.csv file created in Log Folder

This is a simple implementation of creating and detecting a PUE Attack by making modifications to primary user detection in CR.