Create and detect a Primary User Emulation (PUE) Attack in Cognitive Radio Networks

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 characteristics, 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.

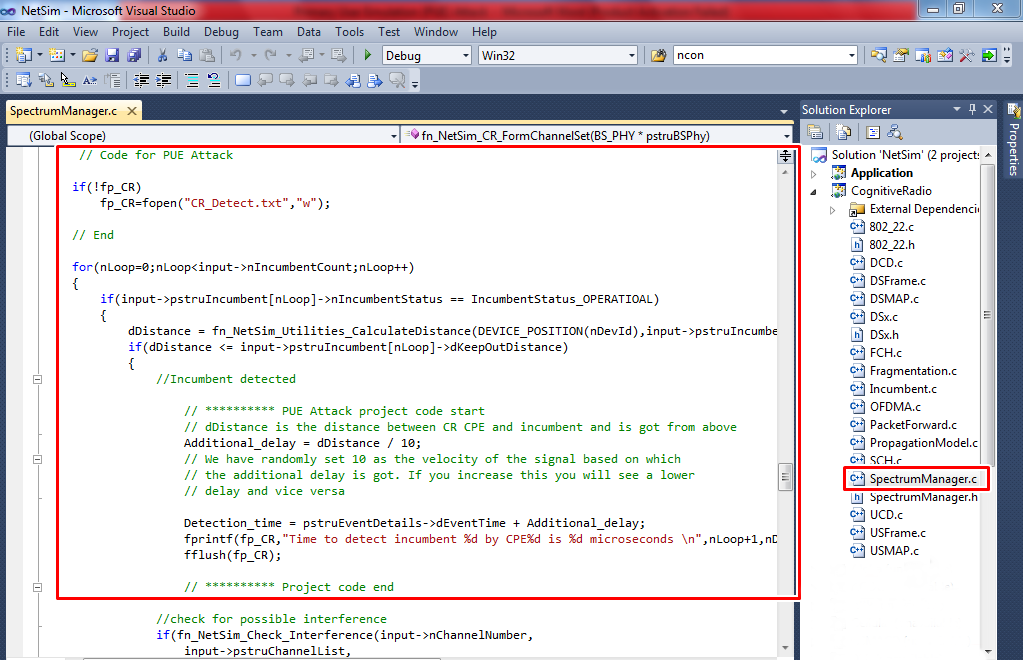
Our next goal is to detect the PUEA by the secondary users. For example purposes we have set the detection time as proportional to the distance of the secondary users from the malicious primary user.

The code given below is for an example implementation of PUE Attack.

# Steps to run Primary User Emulation Attack Code in NetSim:

1) Open the Simulation folder inside the extracted Primary User Emulation Attack folder.

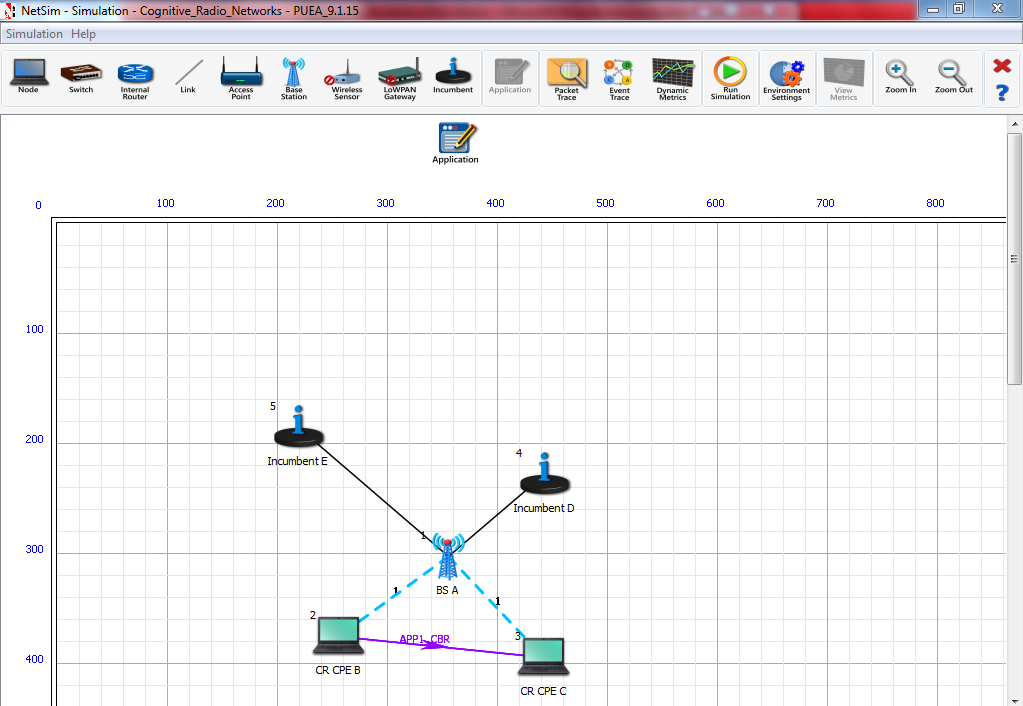
2) Go to CognitiveRadio project 🡪 Open SpectrumManager.c. Inside the **SpectrumManager.c** file, the code to be modified is commented as **PUE Attack code.** Do the required modifications.



3) Now Rebuild Cognitive Radio project.

4) Copy the **libCognitiveRadio.dll** inside **dll** folder and replace it in **bin** path.

5) Now create your scenario in NetSim or you can open the Configuration.xml file that is attached to this zip file.



6) In **CR-BS/Inteface1\_CR/Incumbent** properties, change the value of **Keep Distance = 500** and ensure that the distance between the CPE and Incumbent is <500. This ensures that the incumbent is detected. If the incumbent is beyond the keepout distance then it is not detected.

7) In physical layer, change the **IFQP\_Bitmap** to 1000000000000000.

8) In the Incumbent properties, you can set the values as shown below:

In malicious**, Operational \_Time(s) – 4, Operational interval – 10**

In Incumbent, **Operational \_Time(s) – 9, Operational interval – 9**

The timing diagram is as follows:

Incumbent --- 0s to 9 s (OFF), 9s to 18s (ON), 18s to 27s (OFF), 27s to 36s (ON) ... and so on

Malicious --- 0s to 10s (OFF), 10s to 14s (ON), 14s to 24s (OFF), 24s to 28s (ON) ... and so on

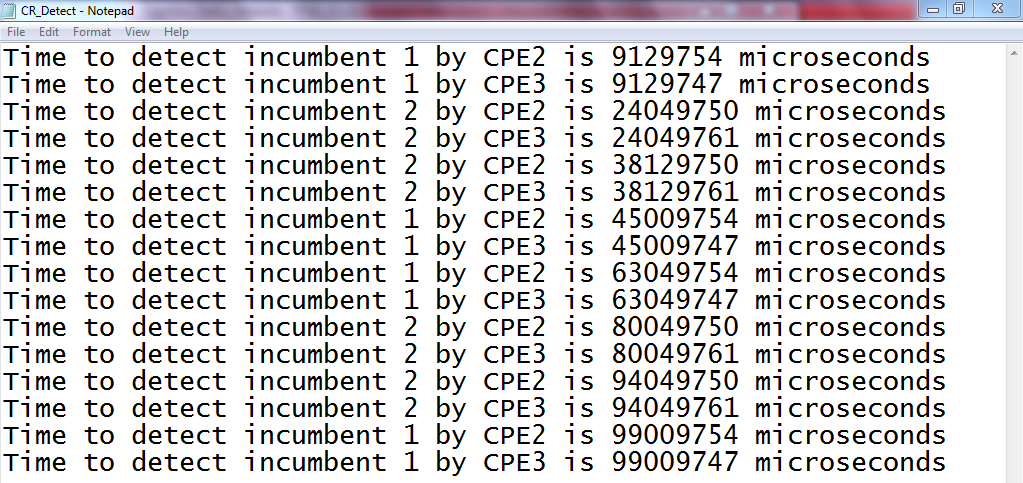
9) Now Run the simulation.

10) You can see the delay in the **CR\_Detect.txt** file inside bin folder. This additional delay has been set by the following code,

**Additional\_delay = dDistance / 10;**

(You can also change the values as 10/100/1000 and analyse different variation in delay.)

A file “**CR\_Detect.txt**” will be created in the bin folder with the following contents:



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