Primary User Emulation (PUE) Attack in Cognitive Radio Networks

Software Recommended: NetSim Standard v13.0 (32/64-bit), Visual Studio 2017/2019

Project Download Link:
https://github.com/NetSim-TETCOS/PUE_Attack_v13.0/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.

Steps to simulate

1. Open the Source codes in Visual Studio by going to Your work-> Workspace Options and Clicking on Open code button in NetSim Home Screen window.

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.

```
//Incumbent detected
// ********** PUE Attack project code start
// dDistance is the distance between CR CPE and incumbent and is got from above
Additional_delay = dDistance / 10;
// We have randomly set 10 as the velocity of the signal based on which
// the additional delay is got. If you increase this you will see a lower
// delay and vice versa
Detection_time = pstruEventDetails->dEventTime + Additional_delay;
fprintf(fp_CR, "Time to detect incumbent %d by CPE%d is %d microseconds \n", nLoop + 1,
nDevId, Detection_time);
fflush(fp_CR);
// ********** Project code end
//check for possible interference
```

3. Right click on the Solution in the solution explorer and select Rebuild.

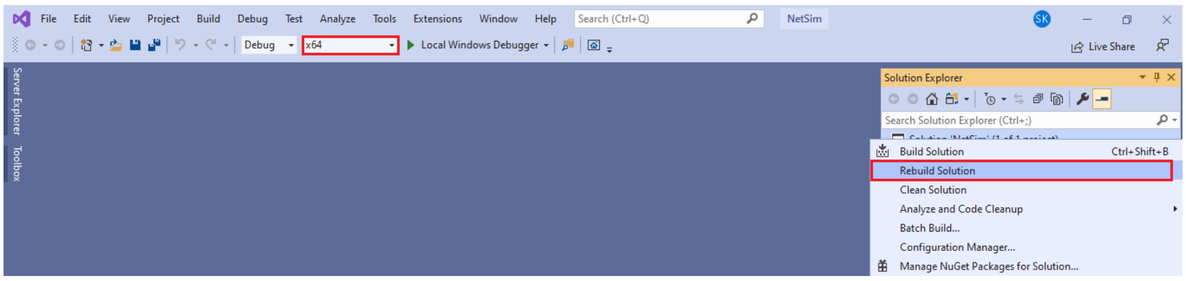


Figure 1: Screen shot of NetSim project source code in Visual Studio

4. Upon successful build modified libCognitiveRadio.dll file gets automatically updated in the directory containing NetSim binaries.

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:

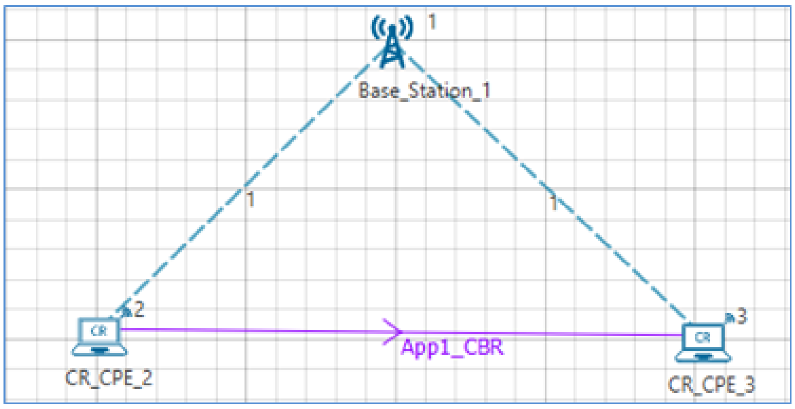


Figure 2: Network Topology

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

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

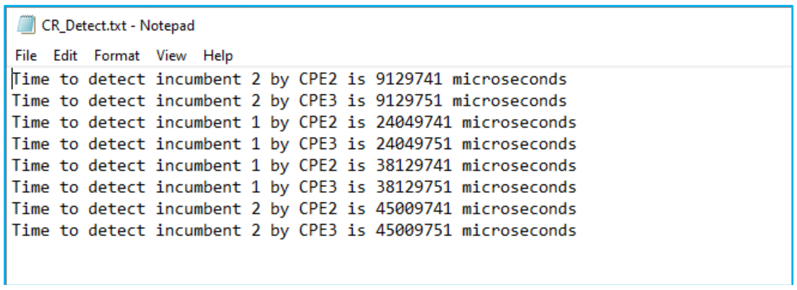
5. In CR-Base_Station_1/INTERFACE_1 (COGNITIVE_RADIO) under physical layer, the IFQP_Bitmap is set to 1000000000000000.
6. Now run the simulation 50 Sec.

Results and discussion

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 analyses different variation in delay.)

A file "**CR_Detect.txt**" will be created in the bin folder (NetSim installation directory) with the following contents:



```
Time to detect incumbent 2 by CPE2 is 9129741 microseconds
Time to detect incumbent 2 by CPE3 is 9129751 microseconds
Time to detect incumbent 1 by CPE2 is 24049741 microseconds
Time to detect incumbent 1 by CPE3 is 24049751 microseconds
Time to detect incumbent 1 by CPE2 is 38129741 microseconds
Time to detect incumbent 1 by CPE3 is 38129751 microseconds
Time to detect incumbent 2 by CPE2 is 45009741 microseconds
Time to detect incumbent 2 by CPE3 is 45009751 microseconds
```

Figure 3: CR_Detect.txt file created in NetSim installation directory

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