

Primary User Emulation (PUE) Attack in Cognitive Radio Networks

Software Recommended: NetSim Standard v14.0, Visual Studio 2022

Project Download Link:

https://github.com/NetSim-TETCOS/Primary-User_Emulation-v14.0/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 addresses 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. 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 objective is to detect PUE attacks, with detection times proportionate to the distance of secondary users from the malicious primary user.

Real-World Context:

In a cognitive radio network (CRN), which facilitates wireless communication by enabling unlicensed users (secondary users, SUs) to utilize spectrum allocated to licensed users (primary users, PUs), a PUE attack occurs. This attack involves the opportunistic use of spectrum not currently in use by PUs, disrupting the normal functioning of the CRN.

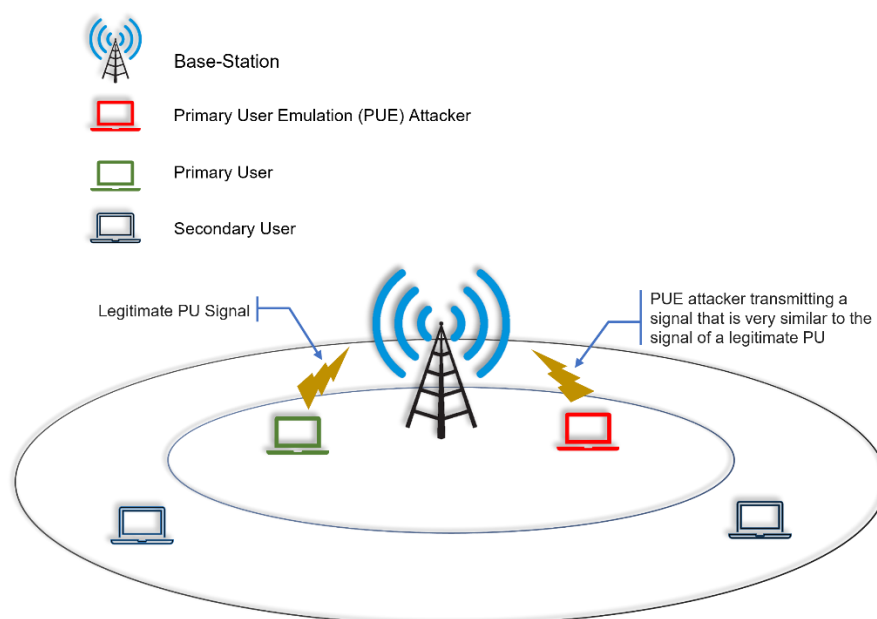


Figure 1: Primary User Emulation(PUE)Attack in Cognitive Radio Networks

PUE attacks are a type of cyberattack where a malicious user impersonates a PU in order to disrupt the communication of SUs. This can be done by transmitting a signal that mimics the characteristics of a PU signal, such as its power level, waveform, and frequency.

The above image shows a PUE attacker transmitting a signal that is very similar to the signal of a legitimate PU. This causes the SUs to believe that a PU is present, even though there is not one. As a result, the SUs vacate the spectrum, even if they are currently using it legitimately.

Primary User Emulation (PUE) Attack in Cognitive Radio Networks overview

- PUE attacker is a malicious user emulating an authorized primary user (PU) to gain unauthorized access to a shared spectrum
- In Cognitive radio networks, where secondary users (SUs) opportunistically utilize licensed spectrum when not in use by Primary users
- The PUE attacker transmits a signal mimicking the characteristics of a Legitimate PU signal
- The secondary users can detect the PUE attack by observing the delay in the detection of the malicious primary user.

The Role of NetSim Simulator:

In the simulation of a Primary User Emulation (PUE) Attack using NetSim, NetSim serves as the essential platform for modeling Cognitive Radio Networks (CRNs). It enables users to configure network parameters, simulate scenarios involving PUE attacks, and analyze the network's response. NetSim's result dashboard and log files provide valuable insights into how secondary users, represented by CR-CPE devices, adapt to dynamic spectrum conditions, and detect potential PUE attacks. This simulation-driven approach within NetSim aids in studying and validating counter measures against security threats in CRNs

Note: In NetSim, primary users are configured within the base station itself and are referred to as incumbents.

Example

1. The **PUE_Attack_Workspace** comes with a sample network configuration that is already saved. To open this example, go to Your work in the Home screen of NetSim and click on the **PUE-Attack-Example**.
2. The network scenario loads as shown below:

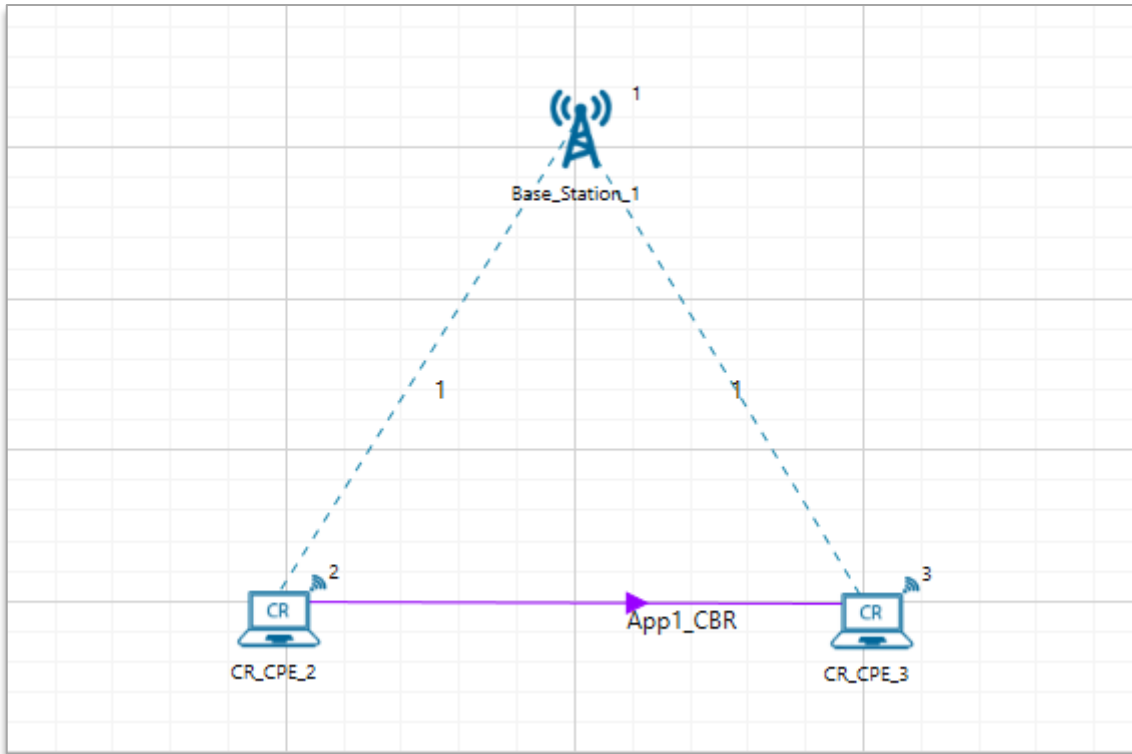


Figure 2: Network Topology

Settings done in this example

- 3. The following properties are set for **Base_Station_1** → **INTERFACE_1 (COGNITIVE_RADIO)** → **IEEE802.22** as shown in below given table.

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

Table 1: CR Base Station 1 Properties

The timing diagram is as follows for **Incumbent1(malicious)** and **Incumbent2(Incumbent)**:

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

- 4. Distance between the CR-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.

5. Now run the simulation 50 Sec.

Note: If the NetSim Simulation Console window halts after completing the simulation, manually terminate it by pressing Ctrl+C until it closes.

Results and discussion

You can see the delay in the **CR_Incumbent_log** file from the log files in the result dashboard Window.

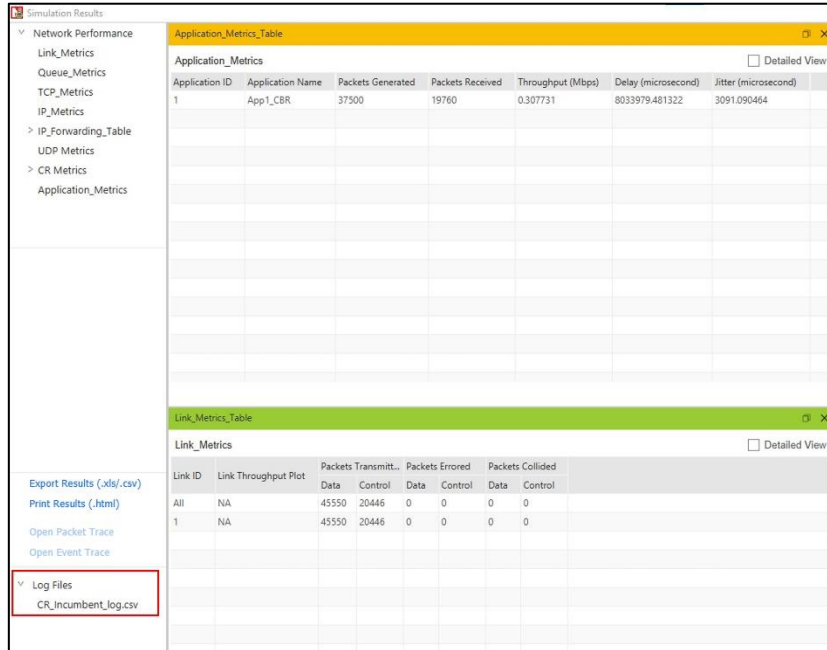


Figure 3: Network Topology

A file “**CR_Incumbent_log**” will be created in the log folder with the following contents: In the “**CR_Incumbent_log.csv**” file we can see secondary users CR-CPE 2 and CR-CPE 3 will detect the PUE attack by Incumbent1 which is the malicious user

CR-CPE ID	Time(Sec)	Incumbent	Device Type
2	9.129741		2 Legitimate PU
3	9.129751		2 Legitimate PU
2	24.049742		1 Malicious PU
3	24.049751		1 Malicious PU
2	38.129742		1 Malicious PU
3	38.129753		1 Malicious PU
2	45.009739		2 Legitimate PU
3	45.00975		2 Legitimate PU

Figure 4: CR_Incumbent_log.csv file created in Log Folder

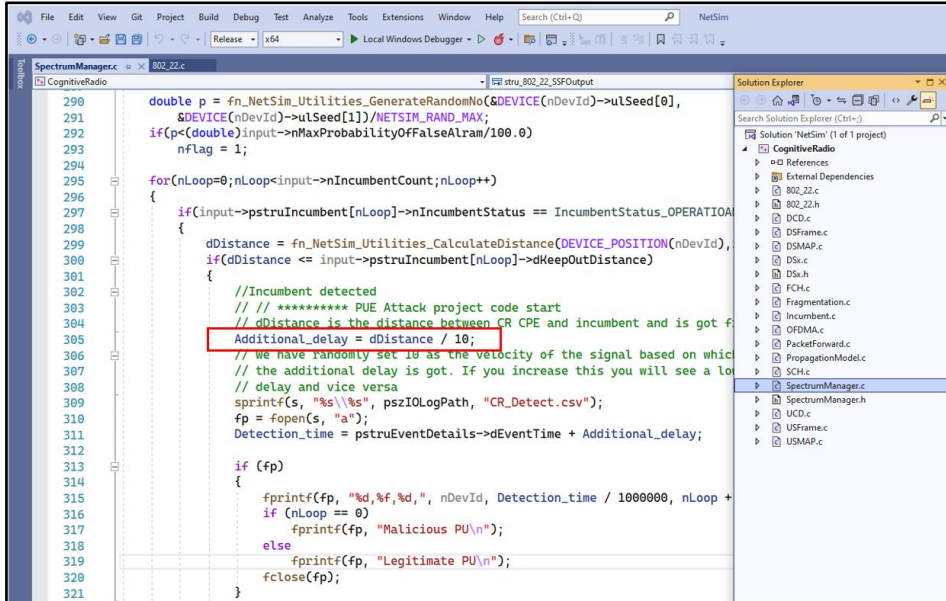
Appendix: NetSim source code modifications and steps.

1. Open the Source code in Visual Studio by going to Your work-> Source Code and Clicking on Open code button.
2. Additional delay has been set by the following code:
In the Solution Explorer, Go to **Cognitive Radio > SpectrumManager.c** and open it.

3. User modifications can be made in the `fn_NetSim_CR_CPE_SSF()` function as mentioned below.
4. Right click on Solution Explorer > Rebuild project.

Additional_delay = dDistance / 10;

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



```

290 double p = fn_NetSim_Uilities_GenerateRandomNo(GDEVICE(nDevId)->ulSeed[0],
291 GDEVICE(nDevId)->ulSeed[1])/NETSIM_RAND_MAX;
292 if(p<(double)input->nMaxProbabilityOfFalseAlarm/100.0)
293     nFlag = 1;
294
295 for(nLoop=0;nLoop<input->nIncumbentCount;nLoop++)
296 {
297     if(input->pstruIncumbent[nLoop]->nIncumbentStatus == IncumbentStatus_OPERATIOA
298     {
299         dDistance = fn_NetSim_Uilities_CalculateDistance(DEVICE_POSITION(nDevId),
300             if(dDistance <= input->pstruIncumbent[nLoop]->dKeepOutDistance)
301         {
302             //Incumbent detected
303             // // ***** PUE Attack project code start
304             dDistance is the distance between CR CPE and incumbent and is got f
305             Additional_delay = dDistance / 10;
306             // we have randomly set 10 as the velocity of the signal based on whic
307             // the additional delay is got. If you increase this you will see a lo
308             // delay and vice versa
309             sprintf(s, "%s\\%s", pszIOLogPath, "CR_Detect.csv");
310             fp = fopen(s, "a");
311             Detection_time = pstruEventDetails->dEventTime + Additional_delay;
312
313             if (fp)
314             {
315                 fprintf(fp, "%d,%f,%d,", nDevId, Detection_time / 1000000, nLoop +
316                 if (nLoop == 0)
317                     fprintf(fp, "Malicious PU\n");
318                 else
319                     fprintf(fp, "Legitimate PU\n");
320                 fclose(fp);
321             }

```

Figure 5: NetSim Project Source Code

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