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

Software Recommended: NetSim Standard v11.0 (64-bit), Visual Studio 2015, 2017
Project Download Link: https://github.com/NetSim-TETCOS/PUEA_v11.0/archive/master.zip

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.

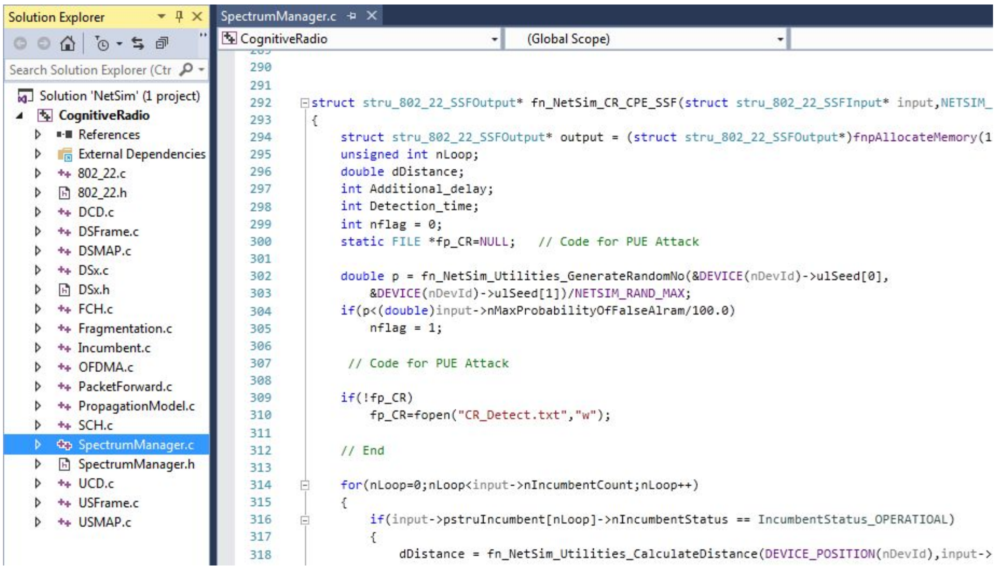
Our next goal is to detect the PUEA by the secondary users. For example purposes we have set the detection time as proportional to the distance of the secondary users from the malicious primary user.

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

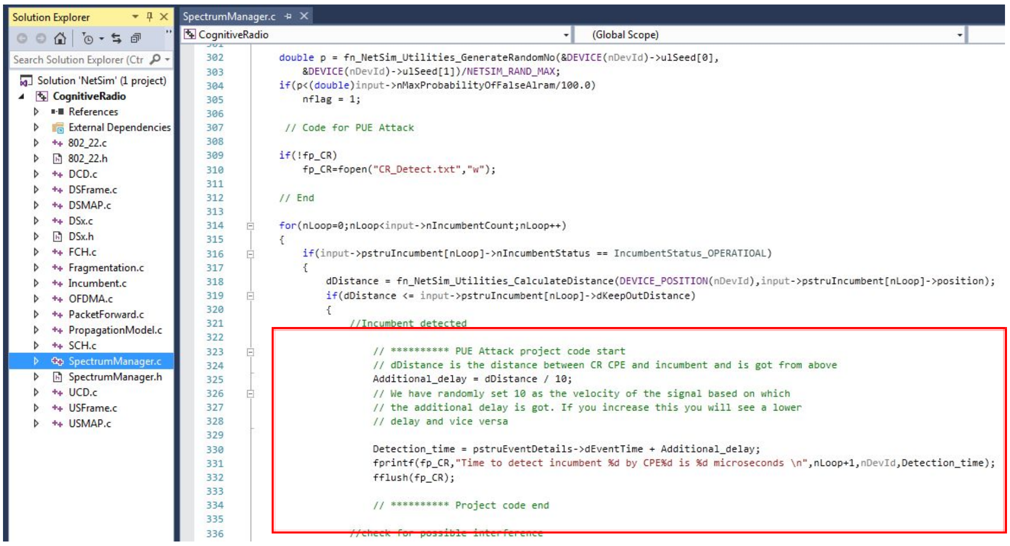
Steps:

1) Open the Code 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.



```
struct stru_802_22_SSFOutput* fn_NetSim_CR_CPE_SSF(struct stru_802_22_SSFInput* input,NETSIM_
{
    struct stru_802_22_SSFOutput* output = (struct stru_802_22_SSFOutput*)fnpAllocateMemory(1
    unsigned int nLoop;
    double dDistance;
    int Additional_delay;
    int Detection_time;
    int nflag = 0;
    static FILE *fp_CR=NULL;   // Code for PUE Attack

    double p = fn_NetSim_Utilities_GenerateRandomNo(&DEVICE(nDevId)->ulSeed[0],
        &DEVICE(nDevId)->ulSeed[1])/NETSIM_RAND_MAX;
    if(p<(double)input->nMaxProbabilityOfFalseAlram/100.0)
        nflag = 1;

     // Code for PUE Attack

    if(!fp_CR)
        fp_CR=fopen("CR_Detect.txt","w");

    // End

    for(nLoop=0;nLoop<input->nIncumbentCount;nLoop++)
    {
        if(input->pstruIncumbent[nLoop]->nIncumbentStatus == IncumbentStatus_OPERATIOAL)
        {
            dDistance = fn_NetSim_Utilities_CalculateDistance(DEVICE_POSITION(nDevId),input->
```

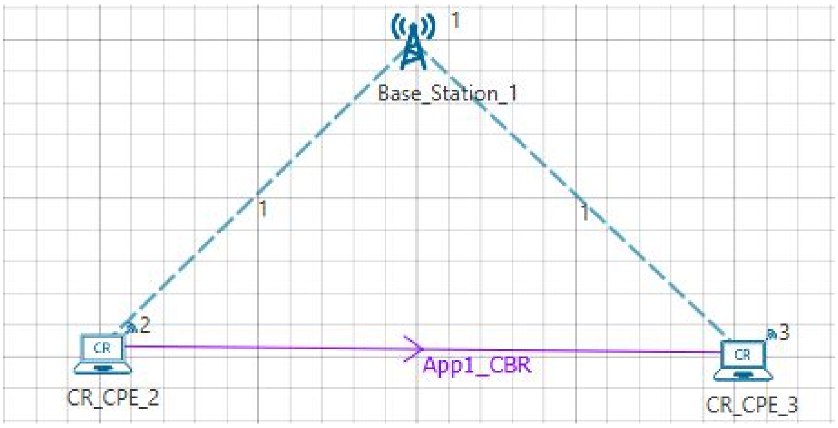


```c
    double p = fn_NetSim_Utilities_GenerateRandomNo(&DEVICE(nDevId)->ulSeed[0],
        &DEVICE(nDevId)->ulSeed[1])/NETSIM_RAND_MAX;
    if(p<(double)input->nMaxProbabilityOfFalseAlram/100.0)
        nflag = 1;

     // Code for PUE Attack

    if(!fp_CR)
        fp_CR=fopen("CR_Detect.txt","w");

    // End

    for(nLoop=0;nLoop<input->nIncumbentCount;nLoop++)
    {
        if(input->pstruIncumbent[nLoop]->nIncumbentStatus == IncumbentStatus_OPERATIOAL)
        {
            dDistance = fn_NetSim_Utilities_CalculateDistance(DEVICE_POSITION(nDevId),input->pstruIncumbent[nLoop]->position);
            if(dDistance <= input->pstruIncumbent[nLoop]->dKeepOutDistance)
            {
                //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) Now Rebuild Cognitive Radio project.

4) Copy the **libCognitiveRadio.dll** from **DLL** folder and replace it in **bin** path i.e **"C:\Program File\NetSim Standard\bin"**.

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



6) In **CR-Base_Station_1/INTERFACE_1 (COGNITIVE_RADIO) Incumbent** properties, Set the **Incumbent count as 2**

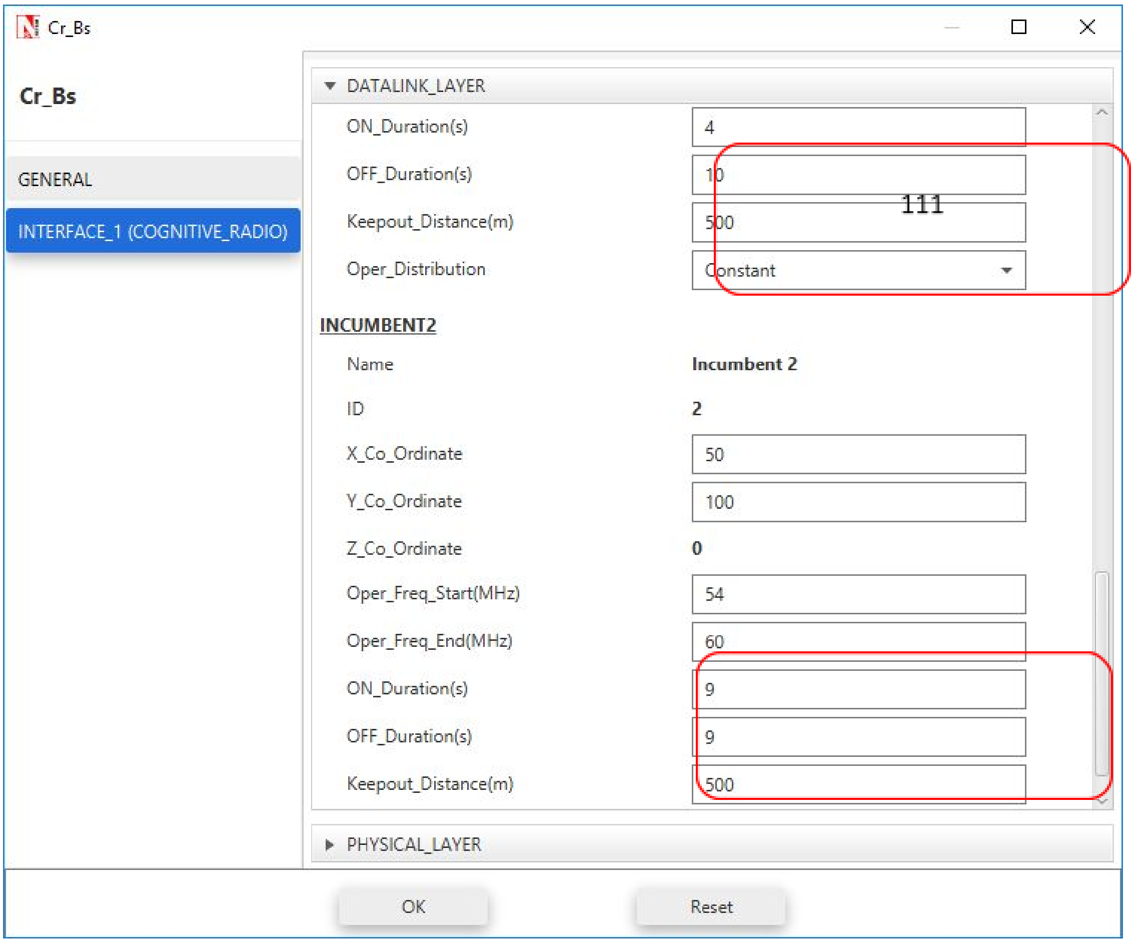
7) In the Incumbent properties, you can set the values as shown below:

In malicious (Incumbent_1), **Operational _Time(s) – 4, Operational interval – 10**
In Incumbent (Incumbent_2), **Operational _Time(s) – 9, Operational interval – 9**
Change the value of **Keep Distance = 500m in both incumbent** 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 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

8) In physical layer, change the **IFQP_Bitmap** to 1000000000000000



Cr_Bs

Cr_Bs

GENERAL

INTERFACE_1 (COGNITIVE_RADIO)

DATALINK_LAYER

ON_Duration(s)	4
OFF_Duration(s)	10
Keepout_Distance(m)	500
Oper_Distribution	Constant

111

INCUMBENT2

Name	Incumbent 2
ID	2
X_Co_Ordinate	50
Y_Co_Ordinate	100
Z_Co_Ordinate	0
Oper_Freq_Start(MHz)	54
Oper_Freq_End(MHz)	60
ON_Duration(s)	9
OFF_Duration(s)	9
Keepout_Distance(m)	500

PHYSICAL_LAYER

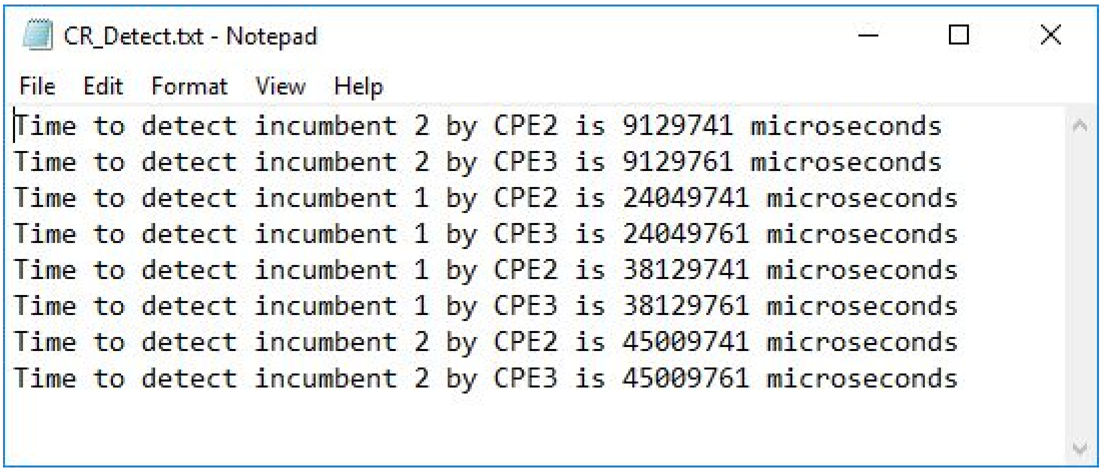
9) Now run the simulation 50 Sec.

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:



```
Time to detect incumbent 2 by CPE2 is 9129741 microseconds
Time to detect incumbent 2 by CPE3 is 9129761 microseconds
Time to detect incumbent 1 by CPE2 is 24049741 microseconds
Time to detect incumbent 1 by CPE3 is 24049761 microseconds
Time to detect incumbent 1 by CPE2 is 38129741 microseconds
Time to detect incumbent 1 by CPE3 is 38129761 microseconds
Time to detect incumbent 2 by CPE2 is 45009741 microseconds
Time to detect incumbent 2 by CPE3 is 45009761 microseconds
```

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