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

Software Recommended: NetSim Standard v12.0 (32/64-bit), Visual Studio 2017/2019

Follow the instructions specified in the following link to clone/download the project folder from GitHub using Visual Studio:
https://tetcos.freshdesk.com/support/solutions/articles/14000099351-how-to-clone-netsim-file-exchange-project-repositories-from-github-

Other tools such as GitHub Desktop, SVN Client, Sourcetree, Git from the command line, or any client you like to clone the Git repository.

Note: It is recommended not to download the project as an archive (compressed zip) to avoid incompatibility while importing workspaces into NetSim.

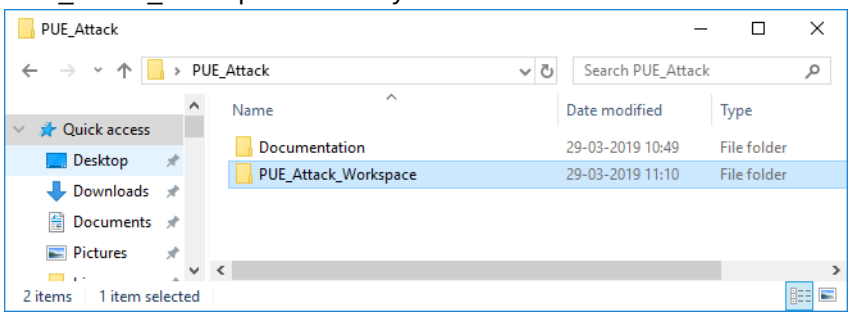
Secure URL for the GitHub repository:

https://github.com/NetSim-TETCOS/PUE_Attack_v12.0.git

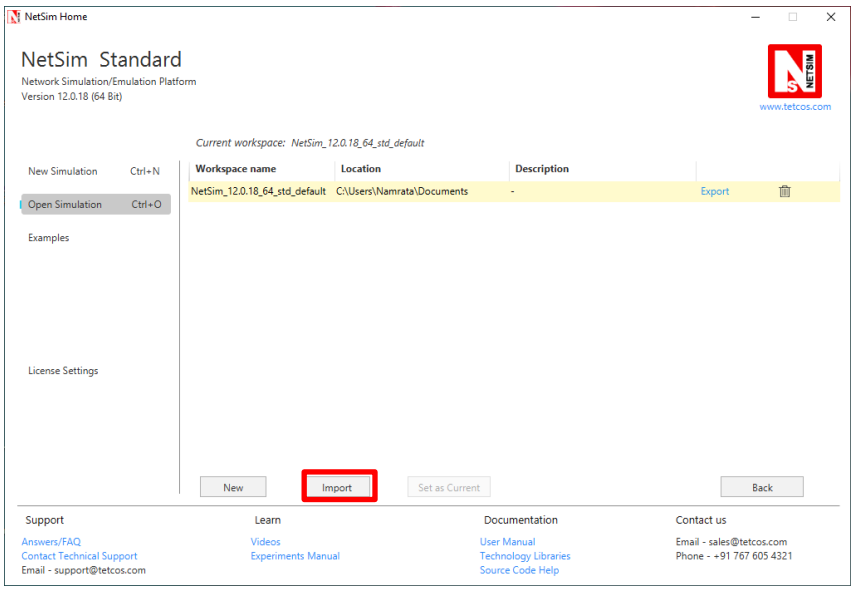
Cognitive Radio (CR) is a promising technology that can alleviate the spectrum shortage problem by enabling unlicensed users equipped with CRs to coexist with incumbent users in licensed spectrum bands while causing no interference to incumbent communications. Spectrum sensing is one of the essential 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:

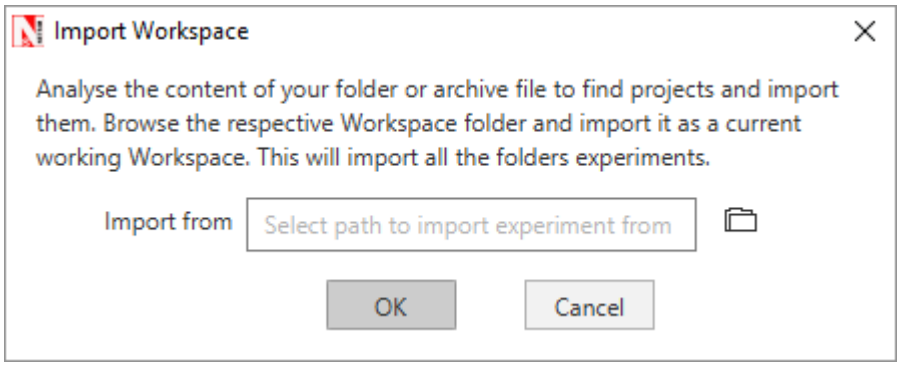
1. The downloaded project folder contains the folders Documentation and PUE_Attack_Workspace directory as shown below:



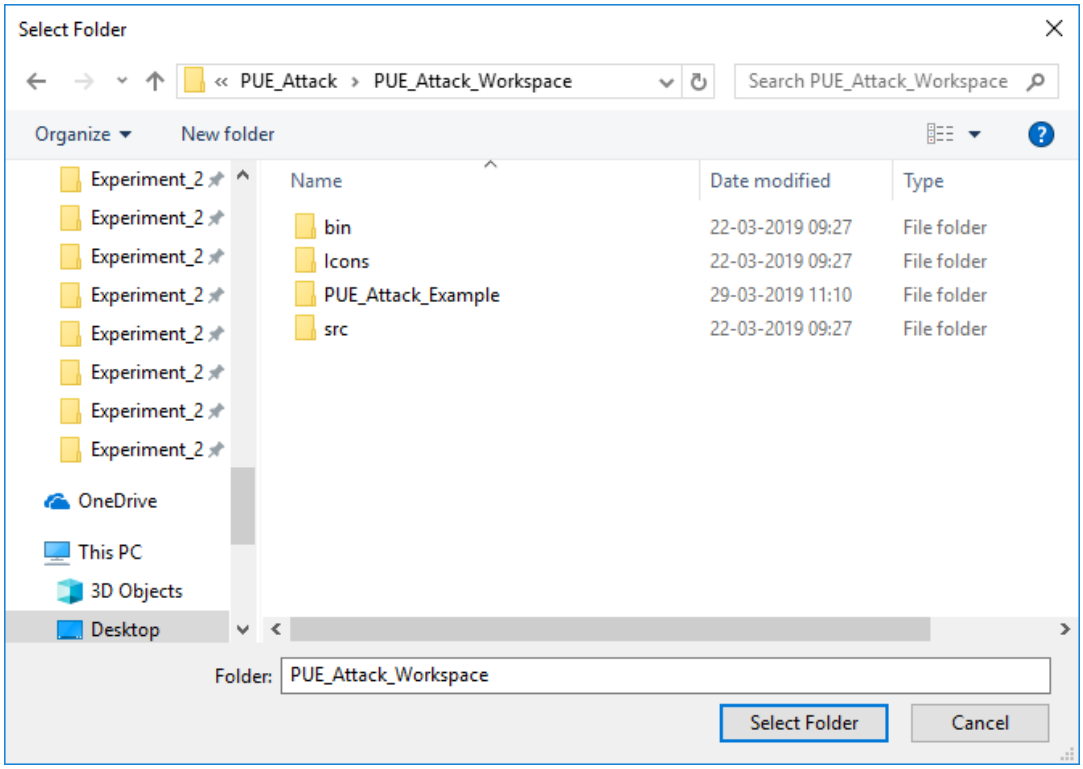
2. Import PUE_Attack_Workspace by going to Open Simulation->Workspace Options->More Options in NetSim Home window. Then select Import as shown below:



3. It displays a window where users need to give the path of the workspace folder and click on OK as shown below:

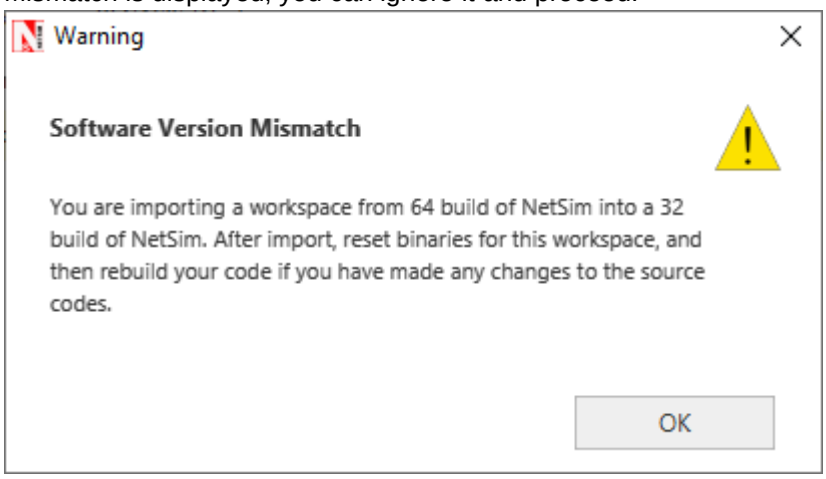


4. Browse to the PUE_Attack_Workspace folder and click on select folder as shown below

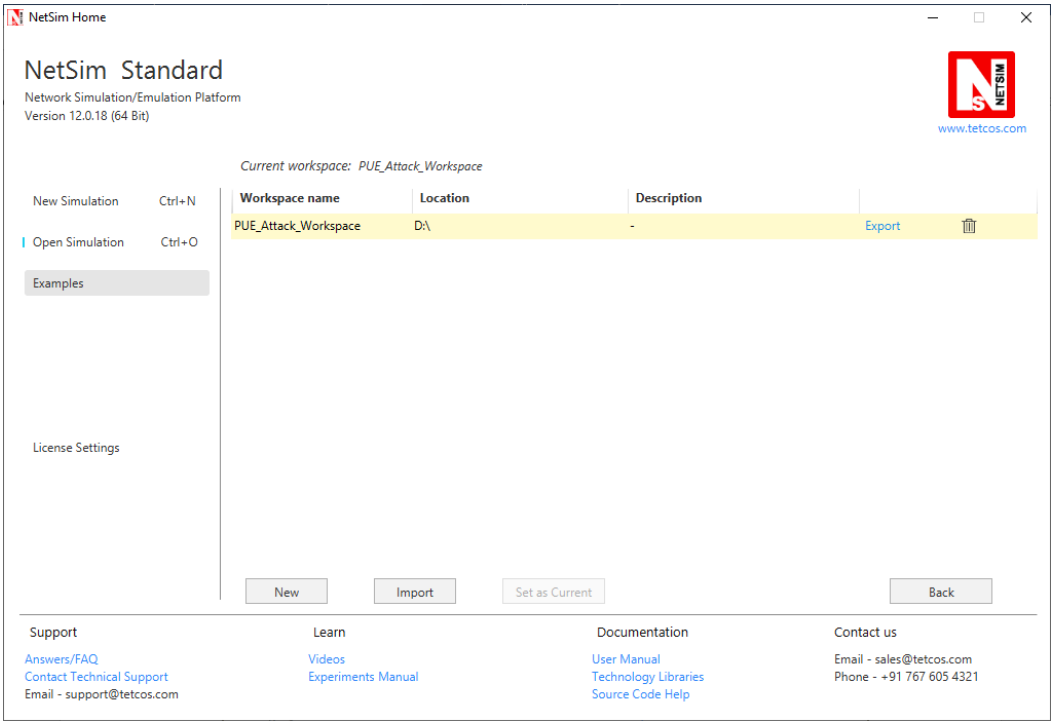


5. After this click on OK button in the Import Workspace window.

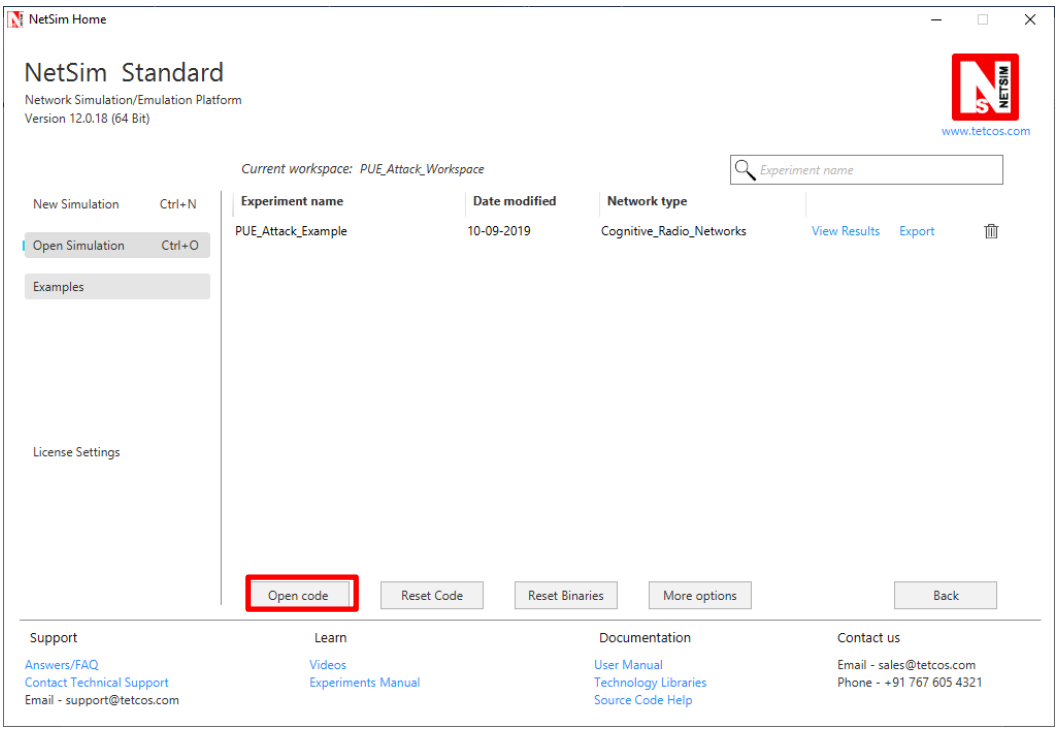
6. While importing the workspace, if the following warning message indicating Software Version Mismatch is displayed, you can ignore it and proceed.



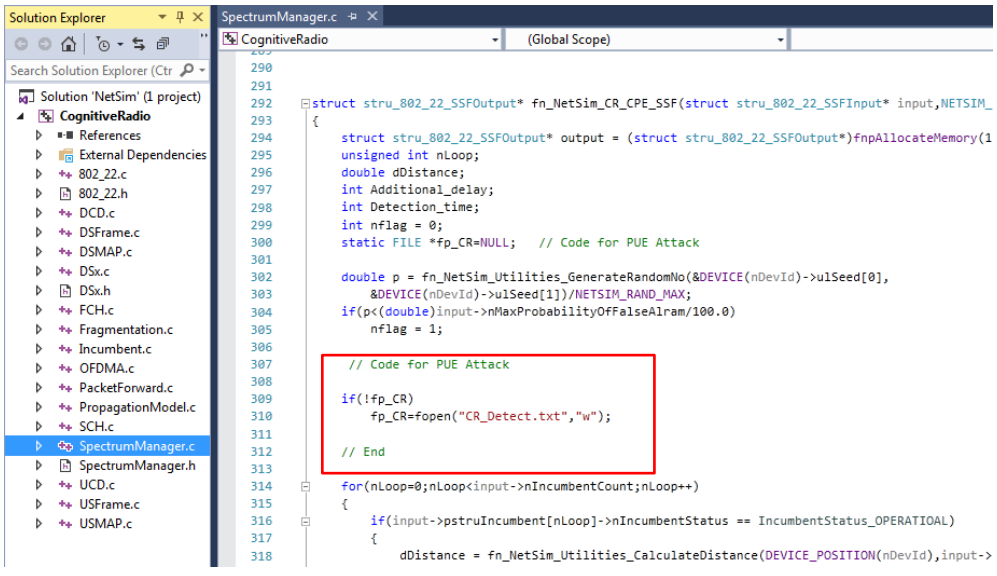
7. The Imported workspace will be set as the current workspace automatically. To see the imported workspace, click on Open Simulation->Workspace Options->More Options as shown below:

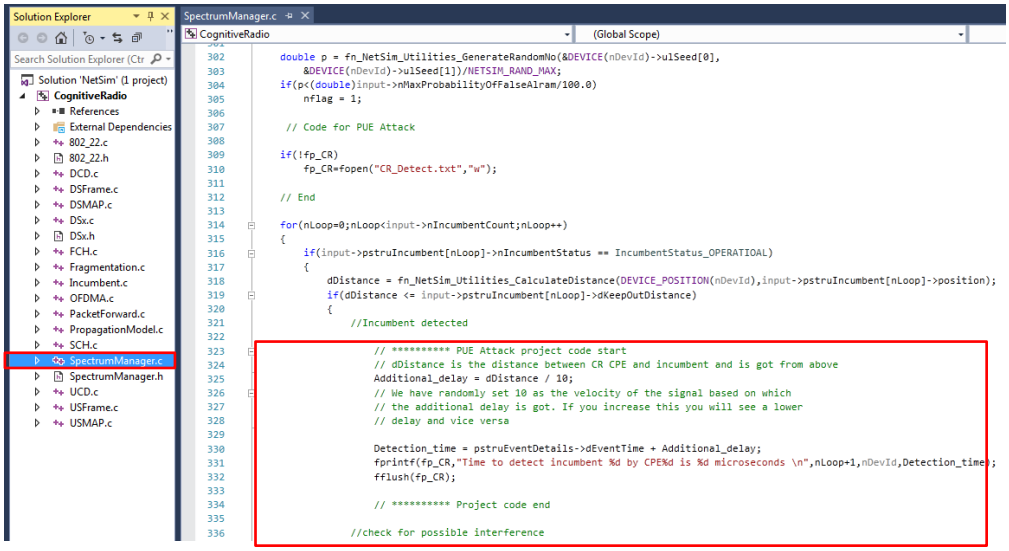


8. Open the Source codes in Visual Studio by going to Open Simulation-> Workspace Options and Clicking on Open code button as shown below:

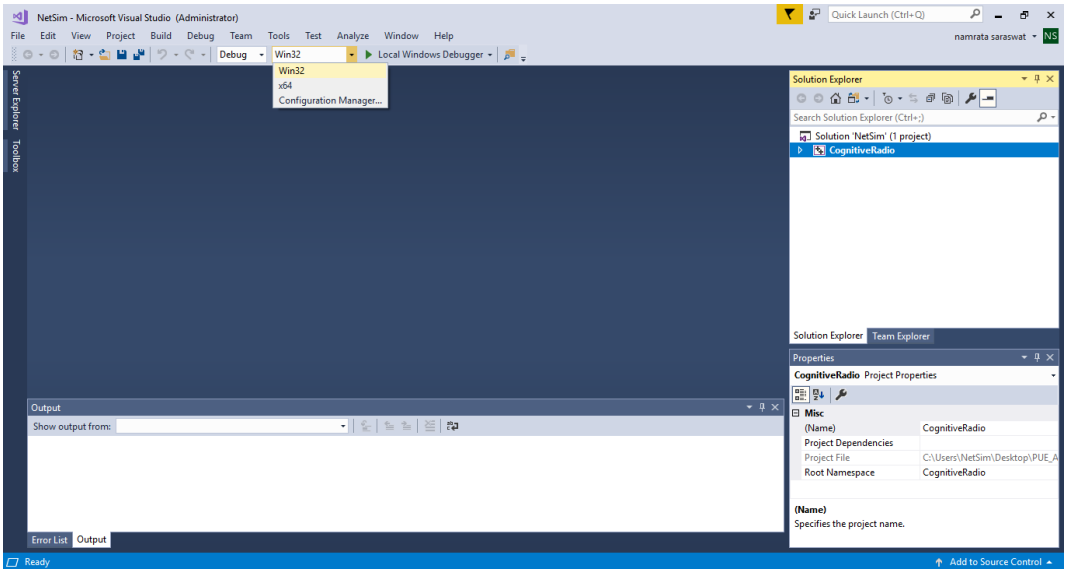


9. Go to CognitiveRadio project → Open SpectrumManager.c. Inside the **SpectrumManager.c** file, the code to be modified is commented as **PUE Attack code.** Do the required modifications.

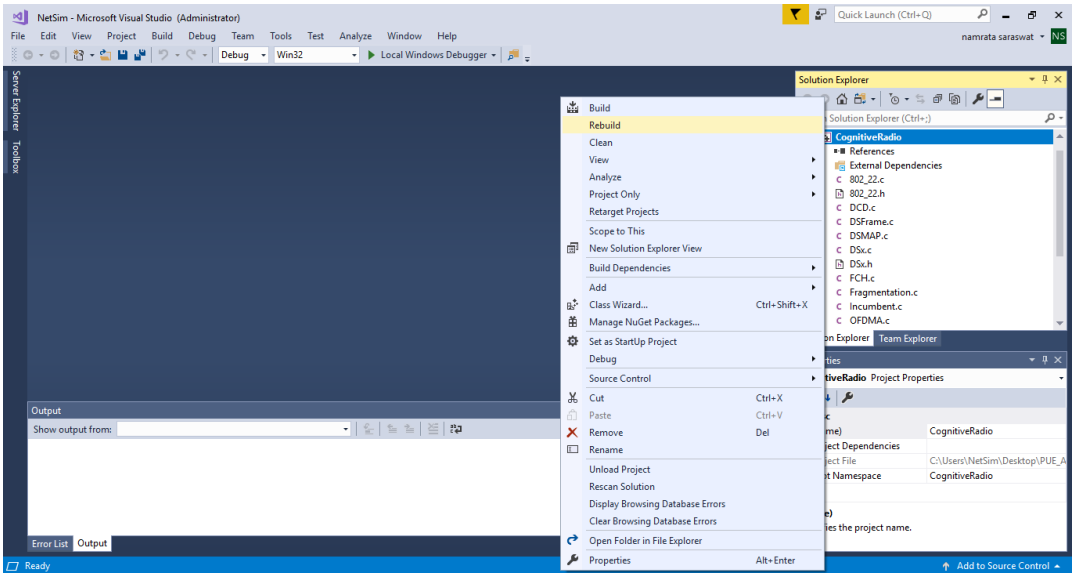




10. Based on whether you are using NetSim 32 bit or 64 bit setup you can configure Visual studio to build 32 bit or 64 bit Dll files respectively as shown below:

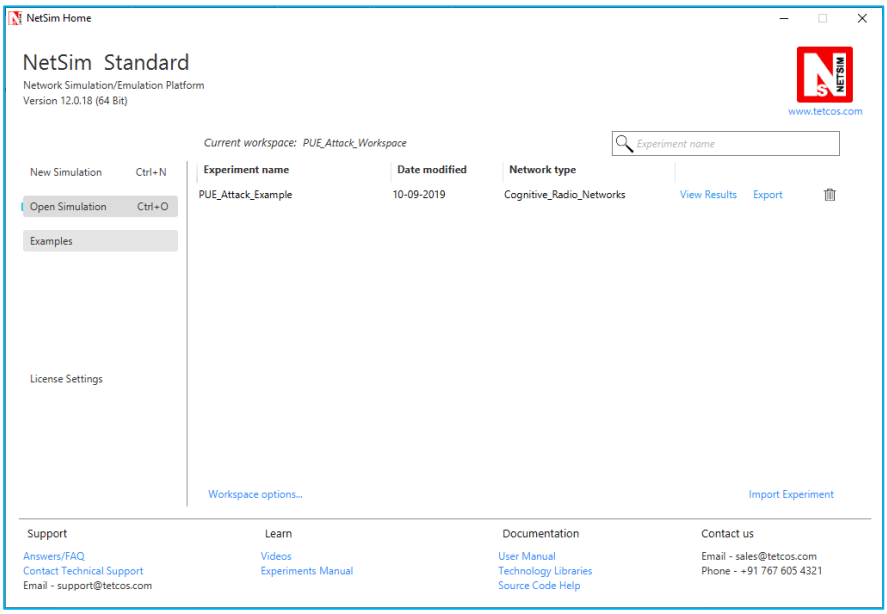


11. Right click on the CognitiveRadio project in the solution explorer and select Rebuild.

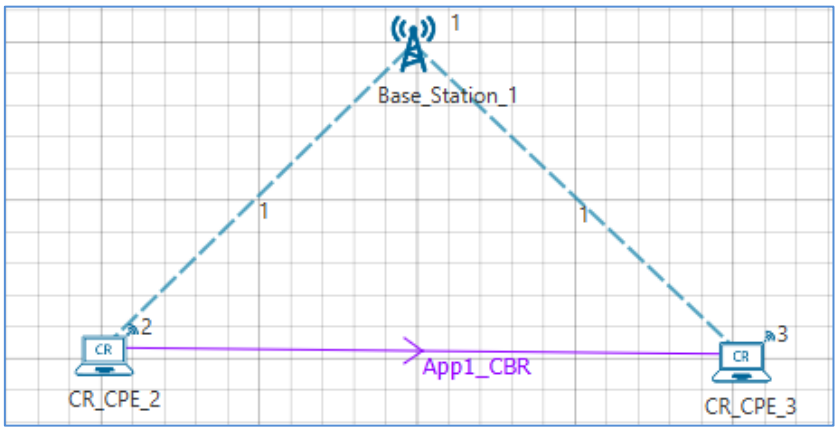


12. Upon successful build modified libCognitiveRadio.dll file gets automatically updated in the directory containing NetSim binaries.

13. Then PUE_Attack_Workspace comes with a sample configuration that is already saved. To open this example, go to Open Simulation and click on the PUE_Attack_Example that is present under the list of experiments as shown below:



14. The network scenario loads as shown below:



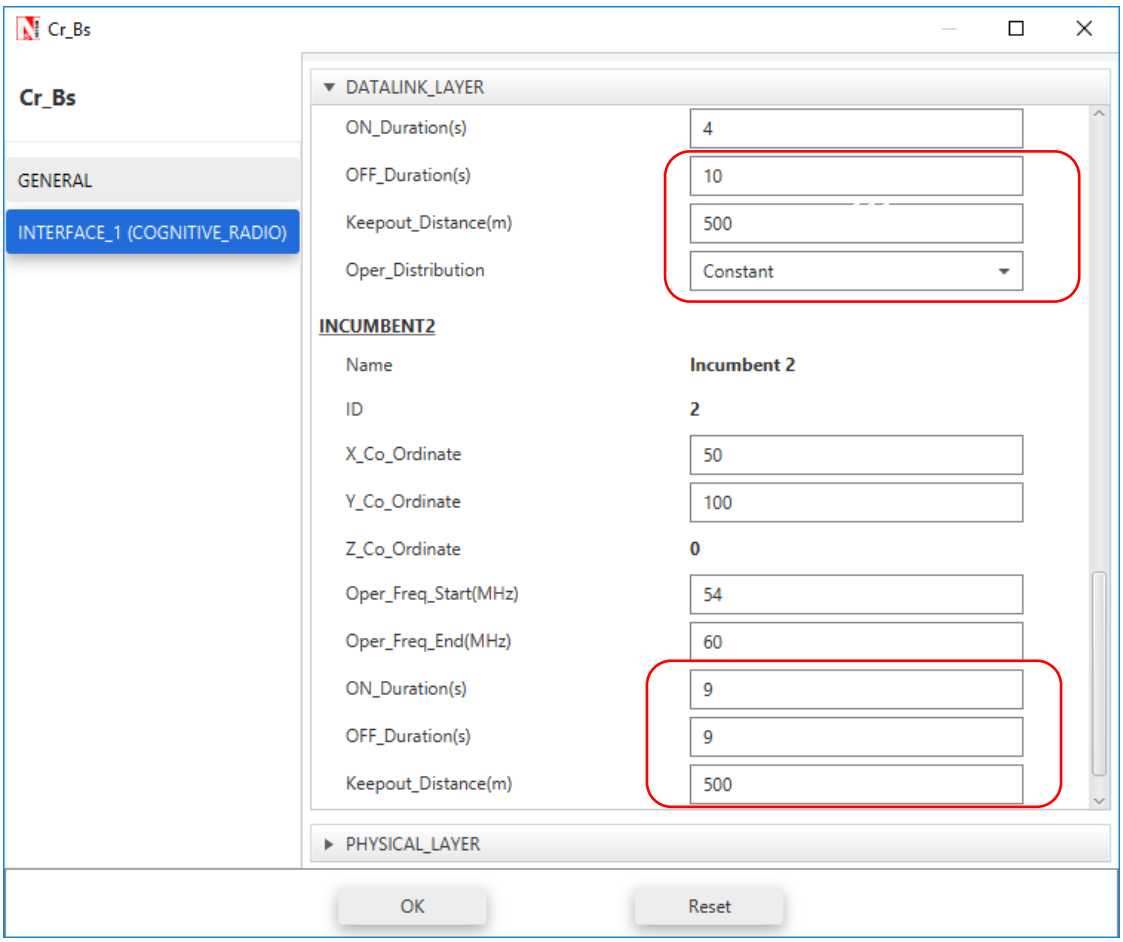
Following settings were done in the devices for this example.

15. In **CR-Base_Station_1/INTERFACE_1 (COGNITIVE_RADIO) Incumbent** properties, the **Incumbent count** is set as **2**
16. In the Incumbent properties:
    In malicious (Incumbent_1)**, ON_Duration(s) – 4, OFF_Duration(s) – 10**
    In Incumbent (Incumbent_2), **ON_Duration(s) – 9, OFF_Duration(s) – 9**
    Keep Distance = 500m in both incumbent and 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

17. In physical layer, the **IFQP_Bitmap** is set to 1000000000000000

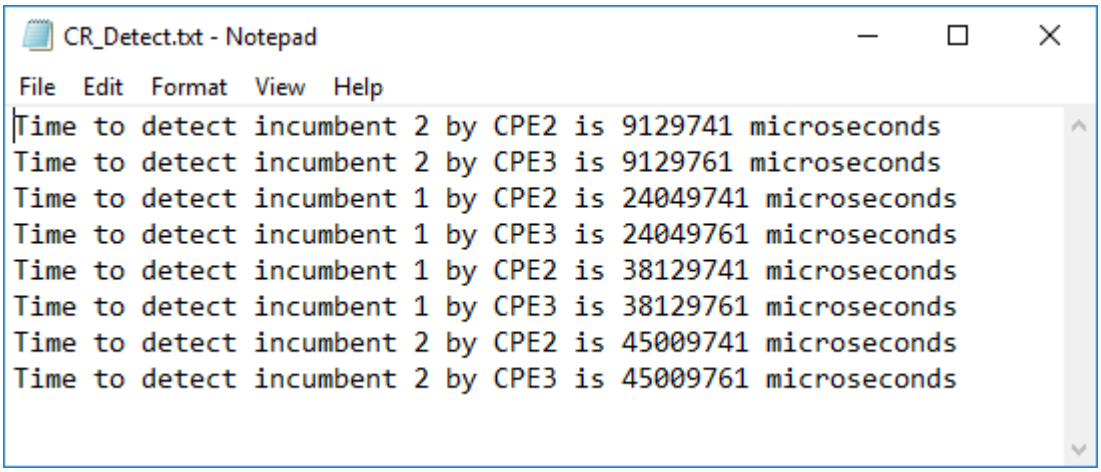


18. Now run the simulation 50 Sec.
19. You can see the delay in the **CR_Detect.txt** file inside bin folder. This additional delay has been set by the following code,

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 (NetSim installation directory) 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.