White Paper

BroadSim How to: Control Power & Pseudorange Offsets of a Repeater Threat

Jaemin Powell

Applications Engineer, Orolia Defense & Security

Introduction

BroadSim's advanced spoofing feature can be a powerful tool to create an environment filled with spoofers and repeaters. Spoofers are used to broadcast signals that look like GNSS signals. These signals can trick a receiver to track the false navigation signal, resulting in the receiver thinking that it is navigating somewhere else. Repeaters receive a broadcasted GNSS signal and rebroadcast the signal to the receiver. This is a true navigation signal, resulting in the receiver thinking that it is at the repeater's receiver position. Both cases can be easily simulated in BroadSim by using the Skydel Simulation Engine.

This paper will outline how to turn on/off spoofers and repeaters, how to adjust their power and how to set a pseudorange offset by providing an example scenario. They will be shown using the manual process by configuring the Skydel instances and the automated process by utilizing Skydel's Python API. If more instruction is needed on how to initialize the spoofer or repeater's output, position or trajectory, please see the "Advanced GNSS Spoofing Simulation Tutorial" video. An automating example using Skydel's Python API is also provided in the "Automating the Advanced GNSS Spoofing Simulation Tutorial" whitepaper.

Scenario Description

In this scenario, the receiver will be stationary, and it will track in a benign environment for the first 10 minutes. Then a repeater threat will power on and try to capture the attention of the receiver. The repeater will be stationary, and the repeater transmitter will be closely located around the receiver. The repeater receiver will be about 1 km from the repeater transmitter. The repeater's signal power will start 3 dB stronger than the truth signal, then it will ramp up by 1 dB every minute until it is 10 dB stronger. After the repeater gets to its max signal strength for a minute, it will turn off and the receiver will end the scenario in a benign environment after 5 minutes.

Skydel Setup

Open the main Skydel instance (Broadsim → Skydel) and in the Settings tab, use the following steps to set up the scenario:

1. Output
    a. Add four DTA-2115B outputs
    b. In Radio 1, edit the Signal Selection to have an Output Type of GNSS, Upper L-Band and a Signal of GPS L1 C/A and Galileo E1.
    c. In Radio 3, edit the Signal Selection to have an Output Type of Interference, check "Choose with signal selection", and select GPS L1 C/A and Galileo E1.
2. Start Time
    a. Select Custom Time box and set the Date to 16 Sep 2020 with a Time of 09:00:00.

3. Vehicle → Body
    a. Set the truth Trajectory to Fixed and Edit the location to:
        i. Latitude = 40.37925159°
        ii. Longitude = -105.50963850°
        iii. Altitude = 2.000 m
4. Spoofers
    a. Click Add Spoofer
    b. Spoofer 1 → General
        i. Click Name box and change the name from "Spoofer 1" to "Repeater 1".
        ii. Uncheck Enabled box.
            1. Please see the **Power On/Off** section for more information on powering on/off spoofers and other signals.
        iii. Edit the Reference Power box to -37.24 dBm.
            1. After completing the **Skydel Set Up** and **Skydel Spoofer Set Up** sections, please see the **Reference Power** section for further explanation on how this reference power was chosen.
    c. Repeater 1 → Trajectory
        i. Set the repeater transmitter's Trajectory to Fixed and Edit the location to:
            1. Latitude = 40.38025854°
            2. Longitude = -105.50522225°
            3. Altitude = 2.000 m

Skydel Spoofer Set Up

The truth signal location and the repeater transmitter location are set up for the scenario. The next step is to set up the repeater receiver location and signals being broadcasted. Open the Skydel Spoofer (1) instance (Broadsim → Skydel Instances → Skydel Spoofer Instance 1) to set up the repeater receiver. In the Setting tab select Output and Add a Spoofer. Edit the Signal Selection and select GPS L1 C/A and Galileo E1 under the Signal section.

After the setup of Output 1 has been completed, both instances will be connected, and time synced to run any scenario. The connection status can be checked in the Skydel instance Settings tab by Spoofers or in the Spoofer submenu (Settings → Spoofers → Repeater 1 → Signal). The status LED by Spoofers, Signal and Spoofer Output Assignation provides a visual of the connection status. Please see the status LED definitions from the Skydel User Manual below:

- Spoofers
  - Gray – The scenario does not contain any spoofer.
  - Red – At least one spoofer's status LED is red.
  - Orange – At least one spoofer's status LED is orange, none are red.
  - Green – All spoofers' status LED are green.
- Spoofers → Repeater 1 → Signal & Spoofer Output Assignation
  - Red – Connection to the spoofing instance could not be established or no signal is configured.
  - Orange – Only some of the spoofer's outputs are assigned to an output.
  - Green – Signals are all assigned to an output.

After the status is in its satisfactory state, both instances are synced together and any scenario can be started from clicking on the Start button in the main Skydel instance. However, since this scenario is using the Skydel Spoofer (1) instance, Start Time and other scenario settings will need to be updated since this is being used as a repeater. The Skydel Spoofer instances are used to transmit any threat signals, including false navigation signals with noncurrent dates and time. This is different than setting up Skydel instances, which are typically used for receiver friendly signals. Based on this, Skydel instances can utilize the master/slave configuration, which will inherit all the scenario settings from the master instance including date and time.

Please follow the next steps in Settings of the Skydel Spoofer (1) instance to complete the set up for the repeater receiver:

1. Start Time
   a. Select Custom Time box and set the Date to 16 Sep 2020 with a Time of 09:00:00.
2. Global → Signal Level
   a. Set Global offset to 3 dB (the signal strength when the repeater turns on).
      i. Please see the **Adjusting Power of Signals** section for a more information on signal level adjustments.
3. Vehicle → Body
   a. Set the repeater receiver's Trajectory to Fixed and Edit the location to:
      i. Latitude = 40.38025750°
      ii. Longitude = -105.51699538°
      iii. Altitude = 2.000 m

4. GPS → Pseudorange Offset → Add...
    a. This will be the distance between the repeater transmitter and the repeater receiver, ~ 1 km.
        i. PRN = All
        ii. Offset = 1000 m
        iii. Please see the **Pseudorange Offset for Time Delay** section for more information on pseudorange offsets.

5. Galileo → Pseudorange Offset → Add...
    a. This will be the distance between the repeater transmitter and the repeater receiver, ~ 1 km.
        i. PRN = All
        ii. Offset = 1000 m
        iii. Please see the **Pseudorange Offset for Time Delay** section for more information on pseudorange offsets.

Automating the Script

After exporting to Python and merging the scripts together (please see step 2 of the Automation Setup section in the "Automating the Advanced GNSS Spoofing Simulation Tutorial" whitepaper for more details on this process), rename the RemoteSpooferSimulator instance to "rep" and replace "sim" in any line that is trying to command the Skydel Spoofer (1) instance in Skydel. The merged script will set up both instances in Skydel. Please see *Figure 1* and *Figure 2* below for a visual of the exports to python for both scripts.

The script has all the API commands to fully automate the scenario (further explanation of the commands in the exports can be found in the **Power On/Off**, **Adjusting Power of Signals** and **Pseudorange Offset for Time Delay** sections). The only steps remaining are to start the simulation, (sim.start()), enable the repeater at 10 minutes (EnableSpoofTx()), increase the repeater signal power every minute from by 1 dB until the signal power is 10 dB stronger than the truth signal (SetPowerGlobalOffset()), disable the repeater a minute after the last signal power increase (EnableSpoofTx()) and stop the simulation 5 minutes later (sim.stop()). Please see *Figure 3* for an example on merging and how to add the steps above to fully automate the scenario.

```python
#!/usr/bin/python

# This Python script has been generated by the SKYDEL GNSS simulator

import ...

sim = RemoteSimulator(True)
sim.connect()

sim.call(New(True, True))
sim.call(SetModulationTarget("DTA-2115B", "", "", True, "{ac1bba84-7c41-48dc-9f85-7939c2f6207b}"))
sim.call(SetModulationTarget("DTA-2115B", "", "", True, "{f62007ae-2a2d-4b75-af9c-e0c3df4b2ae2}"))
sim.call(SetModulationTarget("DTA-2115B", "", "", True, "{ad1a8a95-ddc9-4e5e-b219-bcf9865860f6}"))
sim.call(SetModulationTarget("DTA-2115B", "", "", True, "{95dc6f90-f515-4a9e-aa2b-1cbd3bb8b242}"))
sim.call(ChangeModulationTargetSignals(0, 12500000, 85000000, "UpperL", "E1,L1CA", 50, True, "{ac1bba84-7c41-48dc-9f85-7939c2f6207b}", None))
sim.call(ChangeModulationTargetInterference(0, 12500000, 85000000, 1, 1.57542e+09, 30, "{ad1a8a95-ddc9-4e5e-b219-bcf9865860f6}", "E1,L1CA"))
sim.call(SetGpsStartTime(datetime(2020, 9, 16, 9, 0, 0)))
sim.call(SetVehicleTrajectoryFixEcef("Fix", -1.30105e+06, -4.68836e+06, 4.11016e+06, 0, 0, 0))
sim.call(AddSpoofTx("Spoofer 1", True, "127.0.0.1", 1, "{84172c3a-ac87-4c71-987b-0c38e804fd35}"))
sim.call(RenameSpoofTx("Repeater 1", "{84172c3a-ac87-4c71-987b-0c38e804fd35}"))
sim.call(EnableSpoofTx(False, "{84172c3a-ac87-4c71-987b-0c38e804fd35}"))
sim.call(SetSpoofTxRefPower(-37.24, "{84172c3a-ac87-4c71-987b-0c38e804fd35}"))
sim.call(SetSpoofTxFixEcef(-1.30067e+06, -4.68839e+06, 4.11024e+06, 0, 0, 0, "{84172c3a-ac87-4c71-987b-0c38e804fd35}"))
sim.disconnect()

```

Figure 1: Skydel instance Export to Python.

```python
#!/usr/bin/python

# This Python script has been generated by the SKYDEL GNSS simulator

import ...

sim = RemoteSpooferSimulator(True)
sim.connect()

sim.call(New(True, True))
sim.call(SetModulationTarget("Spoofer", "", "", True, "{83e59096-cf93-477a-90d1-de6852629036}"))
sim.call(ChangeModulationTargetSignals(0, 1250000, 100000000, "UpperL", "L1CA,E1", 0, False, "{83e59096-cf93-477a-90d1-de6852629036}", None))
sim.call(SetGpsStartTime(datetime(2020, 9, 16, 9, 0, 0)))
sim.call(SetVehicleTrajectoryFixEcef("Fix", -1.30163e+06, -4.68812e+06, 4.11024e+06, 0, 0, 0))
sim.call(SetPseudorangeRamp("GPS", 0, 1000, 0, 0, 0, 0, "{28608c25-bdee-4551-bef0-66fee7e81d37}"))
sim.call(SetPseudorangeRamp("Galileo", 0, 1000, 0, 0, 0, 0, "{c552ab7e-b880-4aad-9151-16e83d4a49bb}"))
sim.call(SetPowerGlobalOffset(3))
sim.disconnect()

```

Figure 2: Skydel Spoofer (1) instance Export to Python.

```python
#!/usr/bin/python

# This Python script has been generated by the SKYDEL GNSS simulator

from datetime import datetime
from datetime import date
from skydelsdx import *
from skydelsdx.commands import *

sim = RemoteSimulator(True)
sim.connect()

sim.call(New(True, True))
sim.call(SetModulationTarget("DTA-2115B", "", "", True, "{ac1bba84-7c41-48dc-9f85-7939c2f6207b}"))
sim.call(SetModulationTarget("DTA-2115B", "", "", True, "{f62007ae-2a2d-4b75-af9c-e0c3df4b2ae2}"))
sim.call(SetModulationTarget("DTA-2115B", "", "", True, "{ad1a8a95-ddc9-4e5e-b219-bcf9865860f6}"))
sim.call(SetModulationTarget("DTA-2115B", "", "", True, "{95dc6f90-f515-4a9e-aa2b-1cbd3bb8b242}"))
sim.call(ChangeModulationTargetSignals(0, 12500000, 85000000, "UpperL", "E1,L1CA", 50, True, "{ac1bba84-7c41-48dc-9f85-7939c2f6207b}", None))
sim.call(ChangeModulationTargetInterference(0, 12500000, 85000000, 1, 1.57542e+09, 30, "{ad1a8a95-ddc9-4e5e-b219-bcf9865860f6}", "E1,L1CA"))
sim.call(SetGpsStartTime(datetime(2020, 9, 16, 9, 0, 0)))
sim.call(SetVehicleTrajectoryFixEcef("Fix", -1.30105e+06, -4.68836e+06, 4.11016e+06, 0, 0, 0))
sim.call(AddSpoofTx("Spoofer 1", True, "127.0.0.1", 1, "{84172c3a-ac87-4c71-987b-0c38e804fd35}"))
sim.call(RenameSpoofTx("Repeater 1", "{84172c3a-ac87-4c71-987b-0c38e804fd35}"))
sim.call(EnableSpoofTx(False, "{84172c3a-ac87-4c71-987b-0c38e804fd35}"))
sim.call(SetSpoofTxRefPower(-37.24, "{84172c3a-ac87-4c71-987b-0c38e804fd35}"))
sim.call(SetSpoofTxFixEcef(-1.30067e+06, -4.68839e+06, 4.11024e+06, 0, 0, 0, "{84172c3a-ac87-4c71-987b-0c38e804fd35}"))

rep = RemoteSpooferSimulator(True)
rep.connect()

rep.call(New(True, True))
rep.call(SetModulationTarget("Spoofer", "", "", True, "{83e59096-cf93-477a-90d1-de6852629036}"))
rep.call(ChangeModulationTargetSignals(0, 1250000, 100000000, "UpperL", "L1CA,E1", 0, False, "{83e59096-cf93-477a-90d1-de6852629036}", None))
rep.call(SetGpsStartTime(datetime(2020, 9, 16, 9, 0, 0)))
rep.call(SetVehicleTrajectoryFixEcef("Fix", -1.30163e+06, -4.68812e+06, 4.11024e+06, 0, 0, 0))
rep.call(SetPseudorangeRamp("GPS", 0, 1000, 0, 0, 0, 0, "{28608c25-bdee-4551-bef0-66fee7e81d37}"))
rep.call(SetPseudorangeRamp("Galileo", 0, 1000, 0, 0, 0, 0, "{c552ab7e-b880-4aad-9151-16e83d4a49bb}"))

power_lvl = 3

sim.start()
time = 600
sim.call(EnableSpoofTx(True, "{84172c3a-ac87-4c71-987b-0c38e804fd35}"), time)

while power_lvl < 11:
    rep.call(SetPowerGlobalOffset(power_lvl), time)
    time += 60
    power_lvl += 1

time += 60
sim.call(EnableSpoofTx(False, "{84172c3a-ac87-4c71-987b-0c38e804fd35}"), time)
time += 300
sim.stop(time)

sim.disconnect()
rep.disconnect()
```

Figure 3: Fully automated repeater scenario.

Power On/Off

There are two different commands to power the spoofers/repeaters on or off. The first command is using the Skydel Instance to power on/off all signals outputted by the threat. The second command is using the Skydel Spoofer Instance to power on/off by PRN or code type. Both options are shown below with the manual and the Python API commands.

1. All Signals
    a. Manual
        i. Skydel Instance → Settings → Spoofer → [Spoofer/Repeater Name] → General → **Enabled**
    b. Python API
        i. EnableSpoofTx() – Enables/disables the spoofer transmitter using an enable status and the spoofer transmitter's unique identifier.
        ii. Example: "Automating the Advanced GNSS Spoofing Simulation Tutorial" whitepaper

2. Single PRN or Code Type
    a. Manual
        i. Skydel Spoofer Instance → Settings → [GNSS Type] → **Signals**
        ii. Use the All PRN or PRN specific row to power on/off all RF or select a specific code type.
    b. Python API
        i. EnableRF() – Enables/disables any RF signals based on the GNSS signal type and the PRN (0 for all PRNs).
        ii. EnableSignal() – Enables/disables any RF signals based on the code type and the PRN (0 for all PRNs).
        iii. Example: example_rf_signal_control.py

Reference Power

After both instances have been set up and connected, a Reference Power (Tx) can be chosen to get the acceptable signal power of the repeater. The repeater's signal power needs to be the same as the truth signal power at the receiver's location (-130dBm). To get the Reference Power (Rx) at the receiver to be -130 dBm, go to the Map tab in the Skydel instance. Open the Repeater 1 drop down by clicking on the Repeater 1 box. The Reference Power (Tx) row gives the Reference Power at the repeater transmitter. This value can be set in the Settings tab (Spoofers → Repeater 1 → General → Reference Power). The Reference Power (Rx) row gives the power received at the receiver. Click on the + for a dropdown that shows a categorical breakdown. The Transmitter Antenna Gain, Propagation Loss, Vehicle Antenna Gain, and the Reference Power (Tx) rows need to add up to -130dBm to have the same signal power as the truth signal. Follow the equation below to find the correct Reference Power (Tx) to input:

Reference Power (Tx)

= -130dBm + Transmitter Antenna Gain (dBm)- Propagation Loss (dBm)

+ Vehicle Antenna Gain (dBm)

Note: The Transmitter Antenna Gain model can be changed in Settings → Spoofers → Repeater 1 → Antenna of the Skydel instance. The Propagation Loss is modeled by Skydel and can only be ignored in Settings → Spoofers → Repeater 1 → General of the Skydel instance. The Vehicle Antenna Gain model can be changed by going to Settings → Vehicle → Antenna → Models of the Skydel instance.

Adjusting Power of Signals

There are several commands to control the signal power of the spoofer or the repeater. They can control all signal strengths by settings a Global Offset or they can control individual code type and GNSS signals by setting a Signal Offset.

1. All Signals
    a. Manual
        i. Spoofer Transmitter Reference Power
            1. Skydel Instance → Settings → Spoofers → [Spoofer/Repeater Name] → General → **Reference Power**
        ii. Spoofer GNSS Signal Power
            1. Skydel Spoofer Instance → Settings → Global → Signal Level → **Global Offset**
    b. Python API
        i. SetSpoofTxRefPower() – Sets the reference power (dBm) of the spoofer transmitter using the reference power and the spoofer transmitter's unique identifier.
            1. Example: "Automating the Advanced GNSS Spoofing Simulation Tutorial" whitepaper
        ii. SetPowerGlobalOffset() – Sets the global power offset value for all signals.

2. Specific GNSS Code Type
    a. Manual
        i. Skydel Spoofer Instance → Settings → Global → Signal Level → **Signal Offset**
    b. Python API
        i. SetPowerOffset() – Sets the power offset value for the GNSS signal given in the argument.

Pseudorange Offset for Time Delay

In Skydel, the setup of the spoofers and repeaters use a very similar process. The differences between them are the pseudorange offset and what is transmitted. The pseudorange offset value will be used to simulate the coaxial cable going from the repeater's receiver to the repeater's transmitter. This will simulate the time delay it takes the repeater to receive the broadcasted GNSS signal and transmit it. The spoofer will only transmit a GNSS-like signal. It will broadcast a signal of a different location than the location of the spoofer transmitter. There is only one command needed to control the pseudorange offsets for any or all signals.

1. Any/All Signals
    a. Manual
        i. Skydel Spoofer Instance → Settings → [GNSS Type] → **Pseudorange Offset** → Add
    b. Python API
        i. SetPseudorangeRamp() – Sets the pseudorange offset based on the GNSS type, PRN (O for all PRNs), offset value (m), start time, hold start time (sets the ramping duration to the offset value), hold stop time (sets the ramping duration to no offset), the stop time and the spoofer transmitter's unique identifier.