

As seen in the Lab, the homework assignment will involve making a pseudo-type exposure notification system.

A template file for the broadcast function for the exposure app is provided. How the app should work: **[4 pts]**

1. First get the user's permission to run the app
 - Already implemented
2. Check that all the 'app dependencies' are installed
 - This requires the BTLE app to be installed
 - Add the location where the app was installed to the check_install function i.e. replace the 3 dots with the directory path


```
def check_install():  
    btle_install_location = '/home/.../BTLE' # '/usr/src/BTLE' for DragonOS
```
3. Generate a Temporary Exposure Key
 - Already implemented as in the Lab
4. Save the Temporary Exposure Key in a File named *generated_keys.txt* along with the date and time it was created
 - Write the Temporary Exposure Key to the file *generated_keys.txt*
 - The date and time should have the format dd/mm/yy HH:MM:SS
 - The file *cur_date.py* shows how we can get the time value
 - A sample *generated_keys.txt* file is included with the homework files
 - The function *update_generated_file* can be used to create the file
 - The script starts by determining the current location of the python file and creates a directory named *database_files*. The *generated_keys.txt* file is stored in the *database_files* directory as well as any other files created. The location of the python file is stored as we need to change the directory to access the BTLE application wherever it is installed
5. Start a timer so we know when 24 hours have passed since the generation of the key
 - This can be done using `time.time()`
 - We can store the current time using `start = time.time()`
 - And we can check how much time has passed using `time.time() - start`. The resultant value is in seconds
6. Generate a Rolling Proximity ID (random ID), random MAC address, Encrypted Metadata
7. Start a timer so we know when 15 minutes have passed (i.e. we will change data every 15 minutes)
 - Use the same method mentioned before for timing
8. Send 20 exposure notification packets containing our random ID, MAC and metadata with a spacing of 10 ms on Channel 37
 - See the lab manual for the format and how to use BTLE to transmit these frames
 - `os.popen()` can be used to prevent unwanted output from `btle_tx`
9. Send 20 exposure notification packets containing our random ID, MAC and metadata with a spacing of 10 ms on Channel 38
10. Send 20 exposure notification packets containing our random ID, MAC and metadata with a spacing of 10 ms on Channel 39
11. Keep sending packets on these 3 channels sequentially: 37, 38, 39
12. After 15 minutes change the random ID, MAC address and metadata
 - Remember to reset the timer
13. Send the newly constructed packet on the 3 advertising channels
14. Repeating until the Temporary Exposure Key has to be updated
 - Generate new TE Key
 - Reset timer
 - Save value to the file
 - Generate new random ID, MAC address and metadata with this new TE Key

Notes

- For testing purposes use 1 minute intervals to change the random ID, MAC address and metadata
 - The random ID and metadata will not change in the 1 minute interval therefore use: + `randrange(500)` (in the template) to force the data to change in the testing environment
 - Use 5 minute intervals to change the Temporary Exposure Key
- The process is what is important not the cryptography
- Use the `hcitool lescan` (if it works) or the `bluetoothctl le scan` command to sniff packets and use Wireshark to inspect these packet
- BeaconScope can also be used to check that the packet format is correct
- Feel free to modify the template in any other way
 - For example the timing and number of exposures sent on each channel. This can be modified to find the settings which result in the most packets being captured by a receiver.

Sample Output

```
$ python3 broadcasting.py
Welcome to the Exposure Notification App
[?] Do you consent to the use of this app: y
[+] Today's Temporary Key: 5c4d6d3344bd684b907588657e550c5f
[+] Exposure data
    id: 0ac967e386b6938566324aa0632c9903
    MAC address: dc7d89fb622f
    Metadata: a882297c
[+] Transmitting exposure information...
```

 0ac967e3-86b6-9385-6632-4aa0632c9903
Metadata: 2827102588
(Exposure Notification)
MAC: DC:7D:89:FB:62:2F Packets: 20
Distance: 0.0m RSSI: -33 PPS: 1.2

Bonus [2 pts]

The file `bonus_ble_scan.pcap` contains a BLE scan done with `hcitool`. Four different exposure notifications are present (remember the conditions from the specification). For each of the notifications, state the:

- Random MAC address used
- Rolling Proximity Identifier (random ID)
- Encrypted Metadata value as an integer (as seen in the BeaconScope app)

Due Date: Check Moodle for date

Late Submission Deadline: Check Moodle for date (usually 1 week after due date)

Submission Files: the modified *python script* and if Bonus attempted a *file* containing the answers

Location: Moodle Homework9 link