

Weekly Blog 6

Latest

commit:

<https://github.cs.adelaide.edu.au/2020-Mobile-and-Wireless-Systems/CovidGuard-F/commit/ff572884927618df47476d96e44c24b1859ddec9#diff-5d10c3ccf41b5d812b58644688ba5fa1R95>

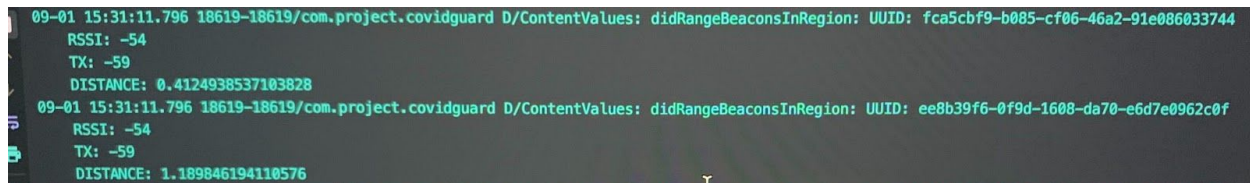
This week I implemented the remaining part of Altbeacon for receiving BLE payloads, which are of the exposure notification layout. I also tested out the payload data elements such as:

1. RPI sent from another device
2. Power of transmission
3. The distance at which the transmission occurred.

All these were derived using the inbuilt functions of the Altbeacon like

1. getId1() for getting the RPI
2. getTxPower() for getting the power of transmission
3. getDistance() for getting the distance at which the transmission occurred.

Sample screenshot:



```
09-01 15:31:11.796 18619-18619/com.project.covidguard D/ContentValues: didRangeBeaconsInRegion: UUID: fca5cbf9-b085-cf06-46a2-91e086033744
RSSI: -54
TX: -59
DISTANCE: 0.4124938537103828
09-01 15:31:11.796 18619-18619/com.project.covidguard D/ContentValues: didRangeBeaconsInRegion: UUID: ee8b39f6-0f9d-1608-da70-e6d7e0962c0f
RSSI: -54
TX: -59
DISTANCE: 1.189846194110576
```

The Altbeacon scanner code snippet is integrated as part of the foreground service. Using a BeaconManager which is the Altbeacon class responsible for scanning in a service or an activity, we can start the scan indefinitely in the service. The bind() function is called when the objects have been initialised. Until the bind() has been called, it is not possible to perform the scanning operations.

Finally, we have the onBeaconServiceConnect() which is a call back that is called once beacons are detected through scanning.

A survey of security vulnerabilities in Bluetooth low energy beacons.

The paper explains about the various types of attacks faced by iBeacon (which is very similar in structure and configuration to AtlBeacon: [AltBeacon and iBeacon Similarity](#)). Some of the main types of attacks faced by beacons are as follows:

1. Spoofing:

Spoofing is a type of attack wherein attackers can use sniffing tools to capture UUIDs of beacons and either impersonate the original beacons by creating and transmitting malicious beacons with the same UUID as that of the original. Applications receiving this would think of the received malicious beacon being safe owing to having the UUID of the original one.

Is **GAEN** vulnerable to this?

No, GAEN uses secure pseudorandom generators to create a temporary exposure key which never leaves the user's device. Further, it makes use of HKDF and AES to derive cryptographically secure Rolling Proximity Identifiers. Moreover, these RPIs are constantly rotated every 10 minutes which makes spoofing extremely hard. As such, GAEN is secure against this type of attack.

2. Denial of Service

Denial of service attacks mainly aims at rendering a service or a device non-operational. From a beacon perspective, the attacker could aim to constantly send out beacons to other devices which could lead to battery drain and crash the device.

Is **GAEN** vulnerable to this?

As per one of the GitHub issues concerning the german Corona-warn-app [3], it is possible to flood a recipient device with packets that could ultimately lead to low storage. More specifically, packets are only transmitted in close proximity which is the underlying principle of BLE. However, an attacker who could amplify the signals and flood packets to recipient devices could potentially initiate DOS. GAEN vulnerability is unclear as the RPIs are stored in Google Play Service. In our project, however, we store the packets locally and as such vulnerability to DOS exists.

3. Hijacking:

Kontakt.io [2] explains BLE hijacking as the process of extracting data from BLE beacons that advertises data in an unencrypted format. For example, a person trying to enter a password to connect with a beacon can be hijacked and used by the attacker rendering the password vulnerable and exploitable.

Is **GAEN** vulnerable to this?

GAEN primarily broadcasts RPIs which have already been derived from one-way hash functions and cryptographically secure random generators. An attacker who gets hold of the RPI is essentially useless to him/her as it cannot be exploited in any manner unless he/she has the TEKs that are used to derive RPIs in the first place. In my opinion, GAEN is not vulnerable to this

What was the conclusion of the research and how can this knowledge be applied to the project?

The research helped in understanding the various types of attacks against beacons. It also highlighted some of the defences in place against the attacks. I have answered the defences concerning GAEN implementation which is an opinion based manner. I intend to use this weekly blog and the previous ones while writing the research paper towards the end of the project to improve the trust and transparency of our application.

REFERENCES

1. Tay, H.J., Tan, J. and Narasimhan, P., 2016. A survey of security vulnerabilities in Bluetooth low energy beacons. Carnegie Mellon University Parallel Data Lab Technical Report CMU-PDL-16-109.
2. Kontakt.io. 2020. Beacons Have Been Vulnerable For Too Long.. [online] Available at <<https://kontakt.io/blog/beacon-security/>> [Accessed 5 September 2020].
3. <https://github.com/corona-warn-app/cwa-documentation/issues/309>