Breaking BLE Beacons For Fun But Mostly Profit

Constantinos Kolias
George Mason University
kkolias@gmu.edu

Lucas Copi
Wayne State University
lucas.copi@wayne.edu

Fengwei Zhang
Wayne State University
fengwei@wayne.edu

Angelos Stavrou
George Mason University
astavrou@gmu.edu

ABSTRACT

Bluetooth Low Energy (BLE) Beacons introduced a novel technology that enables devices to advertise their presence in an area by constantly broadcasting a static unique identifier. The aim was to enhance services with location and context awareness. Although the hardware components of typical BLE Beacons systems are able to support adequate cryptography, the design and implementation of most publicly available BLE Beacon protocols appears to render them vulnerable to a plethora of attacks. Indeed, in this paper, we were able to perform user tracking, user behavior monitoring, spoofing as well as denial of service (DoS) of many supported services. Our aim is to show that these attacks stem from design flaws of the underlying protocols and assumptions made for the BLE beacons protocols. Using a clearly defined threat model, we provide a formal analysis of the adversarial capabilities and requirements and the attack impact on security and privacy for the end-user. Contrary to popular belief, BLE technology can be exploited even by low-skilled adversaries leading to exposure of user information. To demonstrate our attacks in practice, we selected Apple's iBeacon technology, as a case study. However, our analysis can be easily generalized to other BLE Beacon technologies.

1. INTRODUCTION

BLE Beacons and related technologies have recently stirred the attention of smartphone vendors and application developers, primarily owed to their high quality object identification and proximity estimation features and their potential for location based services. In fact, BLE Beacons are the building block of the *Physical Web* [13], Google's own vision for the Web of Things [22].

BLE beacon technologies are based on the BLE [6] wireless communication protocol and more specifically its advertising mechanism that relies on the constant broadcasting of packets. BLE Beacon systems are typically comprised by two main components:

Copyright 2017 ACM 978-1-4503-4935-2/17/04 ...$15.00.
http://dx.doi.org/10.1145/3065913.3065923

A Beacon transmitter device - it has no intelligence and simply broadcasts BLE advertising messages that contain a type of identifier (e.g., a number, a set of numbers, or a URL).

A Beacon reader - it constantly monitor for beacons, retrieves the identifier, and measures its Received Signal Strength Indicator (RSSI).

Arguably, the "killer application" of beacon-based systems is Location-based Advertising (LBA) and related services. In a typical deployment scenario, the beaconing is conducted by devices mounted on strategic locations e.g., in front of items of interest [9]. When users come in proximity to these devices, they receive notifications on their BLE-enabled devices e.g., smartphones. Another popular application is micro-location that involves identifying an object's relative position in short distance with higher granularity. Such scenarios assume that beaconing is conducted by portable tags that can be attached to valuable objects, for instance keys or other valuable devices belonging to end-users.

There is a clear value to using BLE technologies for human-centric applications that depend on fine-grained location information. Given that these devices are targeting everyday activities that are heavily dependent on location, there is a clear need to have these devices designed to protect the end-user against privacy attacks. Unfortunately, based on our study, it appears that the vast majority of the existing beacon schemes naively omit the application of protection to the transmitted BLE Beacon messages (i.e., unique identifiers) and other important protocol-specific parameters, which in turn can lead to a range of attacks.

This paper attempts to enumerate and describe possible vulnerabilities of BLE Beacon technologies using the most prevalent one namely, Apple's iBeacon [4] as a test subject for practical attacks. Our results indicate that BLE Beacon enabled systems are susceptible to a variety of attacks that may target the end-user or the adopting corporation including:

Beacon Hijacking - the registration of any of the beacons operated by a specific vendor by a competing vendor. The motive can be an attempt to aggressively advertise their services within the location of the competitor or to simply deny service to a competitor;

User Profiling - the registration of several vendors' beacons by an unrelated malicious vendor, with aim of receiving updates about the user's location. This in turn enables them to not only conduct fine-grained movement tracking, but also progressively deduce the user's habits;

Presence Inference - in certain application scenarios it is possible to directly associate individuals with their emitting beacons, in order to infer their presence in an area or even track their relative movement;

Beacon Silencing - refers to the manipulation of the underlying protocol mechanics to alter the perceived RSSI, in an attempt to cancel the action that would be triggered under normal conditions;

User Harassment - is created when an active attacker manages to cause faster depletion of the reader's resources. The reader is usually the end-user's smartphone device.

It appears that the associated security risks are so critical that several proprietary BLE-based technologies such as Qualcomm's Gimbal [12] and very recently Google's Eddystone [5] have proposed and incorporated certain security features in their flavor of BLE technology schemes. However, Qualcomm's Gimbal is proprietary, closed source technology and Google's Eddystone is still in experimental stages. Both fail to address the existing security issues in their totality. In this paper, we focus our study on the iBeacon technology because it is commercially available and already in use in many real-word cases.

2. THE BLE & IBEACON TECHNOLOGIES

Bluetooth Low Energy (BLE) also known as Bluetooth Smart [6] is a wireless communication standard specifically intended for devices with limited capabilities and energy resources such as the ones used in IoT applications. The fundamental motivation of BLE is to achieve short-lived connection between devices and burst-like data transfers, relying on energy resources as limited as coin-cell batteries.

Any messages sent in the advertising channels can be leveraged for different purposes including the broadcasting of unprotected/non-sensitive data, for example readings of a temperature sensor, to multiple devices at once. BLE beacon schemes capitalize on the BLE advertising mechanism. Data in the advertising messages of BLE have variable length, and are organized into one or more *advertising data structures*. In BLE, the *advertising data structures* are essentially tuples of the form $< Length, Type, Data >$. The structure of a generic advertising message is contained in Figure 1.

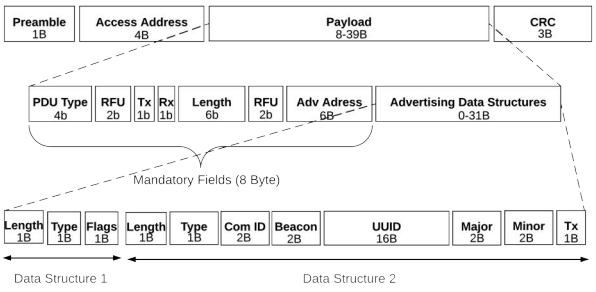

Figure 1: Structure of a BLE Advertising and iBeacon Messages.

While broadcast transmissions are unprotected, BLE applies multiple security mechanisms to protect its connected sessions. Even though the initial pairing mechanism during which the various keys are generated has been the center of critique and is considered insecure [23], we underline the fact that most BLE modules that conform with the protocol in the market are equipped with provably secure cryptographic primitives (AES, HMAC, etc.) to support secure communications.

An iBeacon packet is a BLE advertising packet that is comprised of two predefined *advertising data structures*. Figure 1 contains the structure of the payload of an iBeacon packet. The first structure is mandated by the BLE protocol. It is 3 bytes long and is used to provide values for various flags. The second structure is 26 bytes long and its purpose is to host the various iBeacon identifiers along with other protocol-critical parameters. The most important fields are:
Company ID: In this case it is always the Apple identifier (value 0x00 0x4C);
Beacon Type: The brand of the beacon among the potential existing ones. It is always iBeacon identifier (value 0x02 0x15);
Universally Unique Identifier (UUID): a custom 16-byte number mainly intended to identify the organizations that have deployed the beacons (e.g. a retailer);
Major: a 2-byte number indented to identify the group within which the beacons (say, a specific store of a retailer) are deployed;
Minor: a 2-byte number that identifies a subgroup within which the beacons have been deployed (e.g. a section inside a store);
Measured Power: The estimated received signal strength measured by a receiver that is positioned 1 meter away from the transmitter.

3. THREAT MODEL AND ASSUMPTIONS

The adversary can have various motives depending on her end-goals:

1) Nullify the investment of a competitor by rendering their beacon system unusable; 2) Use existing beacon infrastructures as tools for tracking their valid users or deduce their habits in the mid or long term; 3) Cause annoyance to the end-user of beacon enabled applications that may lead to removal of the corresponding application and hurting the credibility of the corporation.

We assume that the attacker is in possession of software (typically free) and hardware (typically inexpensive) that allows her to sniff BLE packets including beacon messages in larger-than-typical distances. This can be achieved through the use of more efficient high gain antennas. In this way, it becomes possible to capture and replay packets or even craft and inject new ones with fields of choice. We also assume that the specific equipment allows the attacker to broadcast a flood of such packets. In some cases, the adversary must be able to develop a beacon-enabled application and distribute it as legit. Finally, we assume that the attacker can come to proximity with their victim.

In the following section, the specific requirements for the implementation of each attack and its constraints will be further analyzed.

4. PRACTICAL ATTACKS AGAINST BLE BEACONS

4.1 Beacon Hijacking

The term *Beacon Hijacking* refers to the registration of one vendor's beacon-identifiers by another vendor's app, with malicious intentions. While specific identifiers (i.e., UUID, Major, Minor etc.) may be registered and monitored by the benign vendor's app, no technical means exist to prevent other developers from incorporating the same identifiers in their own apps. What is worse, due to the unencrypted nature of the beacon messages, the identifiers associated with an application owner can easily be inferred. Typically, it is used to trigger a malicious action upon sensing the beacons of a competitor, e.g. in scenarios of aggressive advertising campaigns where a notification with an advertising message is displayed right after the valid vendor's message.

The malicious application must pre-register the benign vendor's beacons, thus knowledge of the identifiers associated with the victim is essential. Typically, the number of these identifiers is limited, therefore the adversary can even physically visit each competitor's premises and capture them with the use of appropriate equipment. Any computer with BLE capabilities (including common USB BLE adapters such as the one offered by Plugable [19]), running any Linux based distribution, with Bluez stack [7] suffices for this task.

A major restriction imposed by the iOS platform is the need for explicit location permission since *CoreLocation* services are utilized. Nevertheless, we should underline that while the location permission may discourage some users from installing/using an app, recent studies [8] reveal that almost 35% of the Android apps require the location permission, (b) many apps that practice location tracking are particularly popular, for example, the "Latest Nail Fashion Trends" application has an estimated user base between 100,000 and 500,000, and (c) users are not discouraged to install apps that conduct tracking based on geolocation, even though they have irrelevant purpose e.g., "PokerStars TV" and "Cheezburger".

Another restriction that the iOS platform enforces is the registration of maximum 20-beacon per app, when the monitoring of regions is conducted in the background. At the same time there is a device-wide limit which is however specific to the device model and OS version. Restrictive as these numbers may sound, they are sufficient for most existing scenarios in the retail sectors.

4.2 User Profiling

User Profiling in the context of BLE-beacon, refers to the use of a malicious beacon-enabled application to secretly monitor the user's vicinity for 3rd party beacon messages. Provided that a correlation of the type [*location*, *beacon-identifier*] already exists, the adversary is able to infer the user's position and movement with very high precision i.e., centimeter level.

Essentially, the outcome of this attack is similar to the well-known practice of user location tracking with GPS, although it is preferable for situations where fine-grained positioning is required or for situations where the traditional GPS coverage is not possible (i.e., indoor environments). Through this practice, it is possible to construct a very accurate profile of user's preferences that include details such as how often they visit a retail chain, which departments they prefer, how much time they spend in front of specific products, etc. Such data is invaluable for sectors like personalized advertising.

Similarly to *beacon hijacking*, the adversary must rely on an spyware-like application installed on the user's device to secretly monitor for beacons. The application must report any incident of sensing these beacons to a remote service controlled by the attacker. The difference in this particular attack is that the set of registered beacons is not restricted in one vendor's identifiers, but ideally all possible beacon identifiers should be included.

The biggest challenge for achieving large-scale user surveillance with this method, is the fact that it assumes the existence of a comprehensive database of beacon-identifiers and their corresponding locations/vendors/products. The process for building a database of this kind would be extremely cumbersome for a single attacker, yet online services such as Wikibeacon [21] exist with the purpose of identifying and mapping beacons worldwide.

An additional iOS specific challenge is the limit of 20 registered beacons per app. Indeed, this restriction leaves very little room for effective user tracking. A simple solution for circumventing this limitation would be to rely on GPS positioning to recursively download from a remote web service the expected 20 closest beacons for the wider geographical area of the user and then register these identifiers on the fly. Let us underline that GPS-based positioning and beacon-based proximity estimation rely on the same API services in iOS (CoreLocation), so similarly to *beacon hijacking* only the location permission is required.

The Android Beacon Library enforces no restrictions on the number of regions an application is allowed to monitor, making this attack much more efficient for the Android platform.

Figure 2 displays the web interface of a proof-of-concept tracking system based on beacons that is developed by us. The reader can notice that it is possible to track users with much higher granularity even inside buildings and report on their actions. Tracking of multiple users and filtering according to time is also possible.

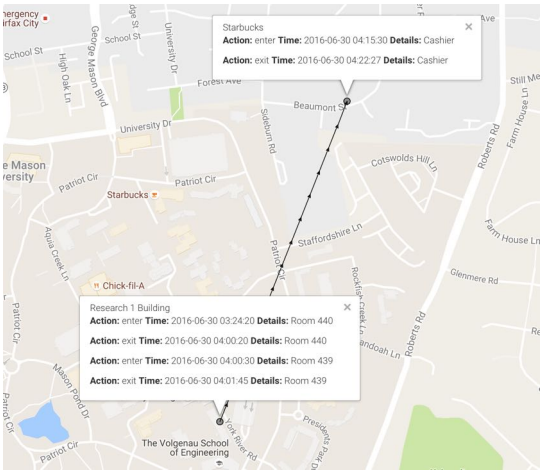

Figure 2: Web interface of beacon tracking application.

4.3 Presence Inference

Presence Inference aims in reporting the presence of a portable, beacon-emitting object that has been associated with a specific user, usually through the means of physical surveillance.

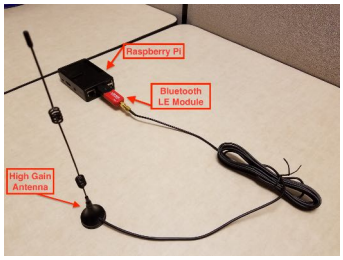

Figure 3: Low cost equipment that can be used for capturing beacon signals.

This vulnerability stems from the fact that the identifiers contained in the beacon messages remain static in every broadcast, and that beacon devices are immutable. Actually, the BLE specification itself attempts to tackle a similar implication by incorporating the concept of private (i.e., randomized) MAC addresses. More specifically, the MAC address that is included in every advertising message of BLE devices gets randomized according to equation 1:

$$MAC = rand||E_{IRK}(rand) \quad (1)$$

where $rand$ is a chosen random number, $||$ is the act of concatenation, and IRK is a derived secret key shared among users and tags. Yet, in the case of BLE Beacon systems even if a similar mechanism would have been applied to randomize the MAC, the existence of fields such as the UUID inside the payload, would defeat the purpose.

Inexpensive hardware equipped with high gain antennas must be placed in strategic locations (e.g., outside supermarkets), and constantly monitor for specific identifiers carried by persons of interest. When the beacon-emitting user passes in front of any monitoring equipment, the event is logged and reported.

The challenges of this approach are associated with the construction of a database of persons/beacon-identifiers. Since this step involves physical surveillance it can not be done in large scale. Thus, this methodology is more appropriate for tracking a limited number of high profile targets. Moreover, due to the difficulty of installing the surveillance equipment in certain areas (e.g., inside a store), it is less effective for profiling the user's behavior or in pinpointing the exact location, but it is more effective for reporting a person's presence among a large crowd (e.g., a protest) or an area.

We conducted a series of experiments to decide the maximum useful distance in which an adversary is able to infer the presence of a beacon inside an area. The equipment used in the experiments consisted of (a) an iPhone 6s Plus, (b) an Android Nexus 4, (c) a laptop equipped with a BLE dongle (Plugable USB Bluetooth 4.0 Low Energy Micro Adapter), as well as (d) the same laptop equipped with a Sena UD100 Long Range Bluetooth 4.0 Class1 USB adapter and a high gain antenna (RP-SMA 2.4GHz 7 DBI). The latter equipment was used to further boost the capture of the signal although it is possible to achieve the same results in a much stealthier fashion by relying on battery powered Raspberry Pi instead of a laptop. The latter assembly is presented in figure 3.

The results indicated that all 3 mobile devices achieve sensing of beacons from a similar range. More specifically, the maximum radius achieved is 74 meters for the laptop, 78 meters for the iPhone 6s Plus and 81 feet for the Nexus 4 device. However, when an equipment with high gain antenna is used, the active radius expands to more than 335 meters. Figure 4 gives a visual indication of the distance of the bearer to the reader when the Android device is used vs the custom high-gain antenna based equipment. In figure 5 a comparison of the coverage areas of the mentioned monitoring nodes is presented. The green circle designates the coverage of the laptop equipment, the purple the iPhone, the blue the Android, while the gray refers to the coverage area achieved with the high-gain antenna equipment.

(a) Android device (b) High gain antenna

Figure 4: Visual comparison of the distance achieved sensing beacon signals.

This practically means that an attacker is able to infer the presence of a beacon-bearing person of interest, inside a large geographic area such as a university campus, by simply placing 3-4 monitoring nodes in strategic locations.

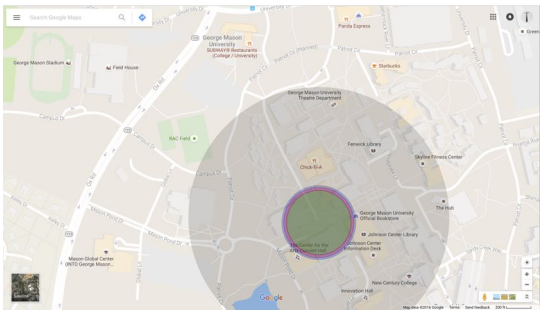

Figure 5: Area covered by various monitoring nodes (ordinary devices cover 75 meters area while devices equipped with high gain antennae cover 300 meters.

4.4 Beacon Silencing

The goal of *Beacon Silencing* is to manipulate a reader into perceiving a beacon tag as remote, while it actually exists within its proximity. This attack is possible primarily due to the exposure of the *Measured Power* field -one of the two variables along with RSSI, that is used in the proximity calculation- as part of every beacon message. An additional source of this vulnerability is the fact that due to the high fluctuations of the RSSI value, the proximity calculation is averaged by multiple signals. For example, in Android Beacon Library the code for estimating the proximity of a beacon tag from the reader is given in Listing 1. Similar methods for the estimation of proximity are defined in iBeacon for iOS and in Eddyston for Android.

Attackers may introduce their own equipment that transmits a flood of spoofed beacons with a greater *Measured*

Power value. In that way, the estimation of proximity will become biased towards the fake readings and result to the calculation of inaccurate proximity estimation.

```
protected static double calculateAccuracy(int txPower,double rssi)
{
   if (rssi == 0) {
     return -1.0; // if we cannot determine accuracy, return -1.
   }

   double ratio = rssi*1.0/txPower;
   if (ratio < 1.0) {
     return Math.pow(ratio,10);
   }
   else {
     double accuracy =  (0.89976)*Math.pow(ratio,7.7095) + 0.111;
     return accuracy;
   }
}
```

Listing 1: Method for estimating the proximity between a beacon tag and reader used in Android Beacon Library.

4.5 User Harassment

This attack of *User Harassment* refers to any action attempting to cause any type of disruption of the normal operation of a valid beacon-enabled reader by exposing it to a flood of forged beacon messages.

Typically, the reader is multi-purposed and multitasking i.e., the beacon-related services it listens to are not the sole purpose of the device as in the case of RFID readers for example. Theoretically, a significant degradation on device's performance could be achieved, mainly due to the large number of filtering operations taking place when a valid reader is exposed to a beacon saturated environment. The reader should take into account that the user's device might have more than one beacon-enabled apps installed, with each application recognizing more than one identifiers. More specifically, for an action to be triggered by the presence of a registered beacon $O(n * a * r)$ checks have to take place, where n is the number of beacons transmitted in the area, a is the number of beacon-enabled apps in the user's device, and r is the number of the registered identifiers recognized by each device. These operations take place constantly per time-segment and may cause faster draining of battery, memory exhaustion, or higher CPU utilization.

Relevant reports [1] indicate that the described process is becoming more efficient due to better utilization of the hardware capabilities in the latest smartphone models. However, our experimental evaluations attest that it is possible to significantly drain the battery of a modern mobile device by transmitting a few iBeacon signals approximately every 30 seconds, if the corresponding application is set to trigger a notification on the entry event. Figure 6 displays the acceleration factor of the energy consumption which reaches 25% on an iPhone 6s Plus (an exhaustion of 687 mA approximately), when an attacker actively transmits a single Beacon message for 4 hours in 30 second intervals. Note that in these experiments all network communication (except Bluetooth) was disabled. A drain of such magnitude is likely to cause annoyance to the user.

5. RELATED WORK

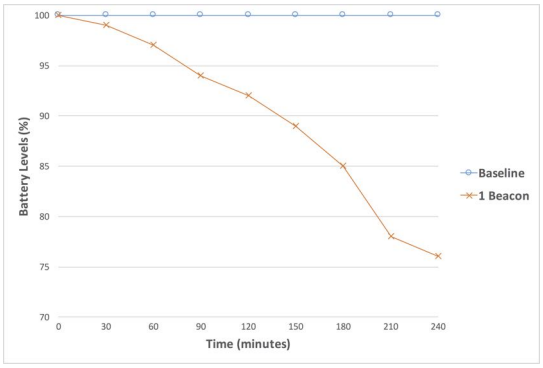

Figure 6: Battery consumption when constantly responding to enter/exit events.

Security in BLE Beacon systems is a heavily neglected topic in research community. The work in [20] was the first one to describe a vulnerability of the iBeacon protocol, i.e., the possibility for the creation of covert channels, to achieve exfiltration of private information from beacon-enabled networks. This vulnerability does not apply in the latest version of the specification.

The bug 67272 of the Android OS was the source for permanently corrupting the Android Bluetooth Service, when the device was exposed to the broadcasting of a Gimbal tag. This behavior (and a potentially vulnerability) was created due to a restriction in the max number of entries (up to 1990) to the database of "seen" Bluetooth MAC addresses. The Gimbal tag frequently changes its MAC address every given interval, thus causing the problematic file to overflow. Reportedly, the bug has been fixed for newer versions of the Android OS.

The authors in [16] identified the privacy risks of commercially available proximity tags and stressed out the lack of protection mechanisms of the transmitted messages.

Researchers Alasdair Allan and Sandeep Mistrymanaged managed to partially reverse-engineer the Estimote iOS SDK and the iOS CoreBluetooth framework using the *method swizzling* technique and the *class-dump* utility [18]. By doing so, the researchers revealed that the Estimote SDK naively used static encryption and decryption keys compromising the smartphone/beacon pairing process. The same researchers were able to hack the iBeacon-enabled promotional scavenger hunt competition at Consumer Electronics Show (CES) [2], [3]. More specifically, by decompiling the corresponding application, the authors were able to infer the UUIDs used for the competition and used them to win the competition from the comfort of their home.

Industrial efforts towards more secure version of beacons include Gimbal [12] and Kontakt.io [17]. These vendors have incorporated their own security-enhanced proprietary protocols that rely on custom rotation mechanisms on the included beacon identifiers. The main aim of these initiatives is to tackle user tracking over prolonged periods of time. On the one hand, tracking is possible between consecutive rotations. On the other, preliminary experiments indicate that static rotation intervals can actually act as beacon indicators defeating the privacy of the schemes. Additionally, such mechanisms seem to have tremendous impact on the exhaustion of the energy resources of the beaconing devices.

In 2016 researchers from Google published a white paper introducing the concept of cloud-based Ephemeral Identifiers as a protection mechanism for the Eddystone platform [14]. Their solution requires symmetric encryption and global synchronization. While the proposed solution tackles the aforementioned vulnerabilities in a satisfactory extend, there are attacks mentioned in the work at hand that are left unaddressed. More specifically, *beacon silencing* and *user harassment*. Moreover, there is skepticism about the scalability of the solution with respect to the cloud service component of the system as well as the energy efficiency of the approach on the beacon-device side.

Privacy threats are not specific to iBeacons, but a large portion of BLE-enabled mobile devices that rely on beacons to advertise their presence permit large-scale tracking. Fawaz et al. [11] conducted an extensive study across 214 BLE-enable devices to conclude that BLE advertising messages, mainly due to unthoughtful design and implementation decisions, leak an alarming volume of artifacts that permits the tracking and fingerprinting of users. Similarly, the study in [10] focuses on BLE-enabled fitness trackers and finds that the majority of them use static hardware address while advertising, which allows user tracking.

6. CONCLUSION

While the advantages of BLE beacon systems are clear they can be overcome by inherent privacy and security risks of such systems. These risks are primarily stemming from the unprotected nature of the BLE Beacon messages. Even though vendors claim that security in the BLE Beacon realm is irrelevant due to the non-mission-critical nature of its application, we have exposed several vulnerabilities that can be exploited by low-skilled adversaries with inexpensive equipment. At the same time we showed that some of the vulnerabilities are violating the end-users' privacy.

several inefficiencies, thus the security of BLE Beacons should be treated as an open research topic.

Our future work will focus on highlighting the shortcomings of such security schemes, along with the development of an efficient security layer on top of an open beacon specification such as the Altbeacon [15] as well as a more efficient security scheme for the Eddystone platform.

References

[1] Aislelabs. iBeacon battery drain on apple vs android: A technical report. http://www.aislelabs.com/reports/ibeacon-battery-drain-iphones/, Last Accessed: 02-12-2017.

[2] A. Allan and S. Mistry. Hacking the CES scavenger hunt. http://makezine.com/2014/01/03/hacking-the-ces-scavenger-hunt/, Last Accessed: 02-12-2017.

[3] A. Allan and S. Mistry. Hacking the CES scavenger hunt for a second time. http://makezine.com/2016/01/07/hacking-ces-scavenger-hunt-second-time/, Last Accessed: 02-12-2017.

[4] Apple. Proximity Beacon specification R1. https://developer.apple.com/ibeacon/, Last Accessed: 02-12-2017.

[5] M. Ashbridge. Eddystone protocol specification. https://github.com/google/eddystone/blob/master/protocol-specification.md, Last Accessed: 02-12-2017.

[6] T. Bluetooth Special Interest Group. Specification of the Bluetooth system, covered core package version: 4.0. https://www.bluetooth.com/specifications/bluetooth-core-specification, Last Accessed: 02-12-2017.

[7] Bluez Project. Bluez official Linux Bluetooth protocol stack. http://www.bluez.org, Last Accessed: 02-12-2017.

[8] B. Botezatu, V. Bordianu, and T. Axinte. Mobile operating system wars – Android vs. iOS. Technical report, Last Accessed: 02-12-2017.

[9] J. Browne. Positioning visitors with iBeacons. https://www.brooklynmuseum.org/community/blogosphere/2014/10/14/positioning-visitors-with-ibeacons/, Last Accessed: 02-12-2017.

[10] A. K. Das, P. H. Pathak, C.-N. Chuah, and P. Mohapatra. Uncovering privacy leakage in BLE network traffic of wearable fitness trackers. In *Proceedings of the 17th International Workshop on Mobile Computing Systems and Applications*, pages 99–104. ACM, 2016.

[11] K. Fawaz, K.-H. Kim, and K. G. Shin. Protecting privacy of BLE device users. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 1205–1221. USENIX Association, 2016.

[12] Gimbal. Gimbal. http://gimbal.com, Last Accessed: 02-12-2017.

[13] Google. The physical Web. https://google.github.io/physical-web/, Last Accessed: 02-12-2017.

[14] A. Hassidim, Y. Matias, M. Yung, and A. Ziv. Ephemeral identifiers: Mitigating tracking & spoofing threats to BLE beacons. 2016.

[15] D. Helms. Altbeacon protocol specification v1.0. https://github.com/AltBeacon/spec, Last Accessed: 02-12-2017.

[16] C. Kolias, A. Stavrou, J. Voas, I. Bojanova, and R. Kuhn. Learning Internet-of-Things security" hands-on". *Security & Privacy, IEEE*, 14(1):37–46, 2016.

[17] Kontakt.io. Kontact.io secure. https://kontakt.io/products-and-solutions/complete-beacon-security/, Last Accessed: 02-12-2017.

[18] S. Mistry and A. Allan. Reverse engineering the Estimote. http://makezine.com/2014/01/03/reverse-engineering-the-estimote/, Last Accessed: 02-12-2017.

[19] Plugable. Plugable usb 2.0 Bluetooth adapter. http://plugable.com/products/usb-bt4le, Last Accessed: 02-12-2017.

[20] J. Priest and D. Johnson. Covert channel over Apple iBeacon. In *Proceedings of the International Conference on Security and Management (SAM)*, page 51. The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp), 2015.

[21] Radius Networks. Wiki Beacon project. http://www.wikibeacon.org, Last Accessed: 02-12-2017.

[22] Raggett, D. Towards the Web of things. www.w3c.org/Talks/0926-dsr-WDC/slides.pdf, Last Accessed: 02-12-2017.

[23] M. Ryan. Bluetooth: With low energy comes low security. In *Proceedings of the 7th USENIX Conference on Offensive Technologies*, WOOT'13, pages 4–4, Berkeley, CA, USA, 2013. USENIX Association.