Covert Channel over Apple iBeacon

Joseph Priest
Rochester Institute of Technology
Rochester NY, USA
joseph@josephjpriest.com

Daryl Johnson
Rochester Institute of Technology
Rochester NY, USA
daryl.johnson@rit.edu

ABSTRACT

Opportunities for covert channels exist using Apple iBeacon technology. Apple iBeacons are an emerging technology designed to provide additional proximity based information to iOS devices. iBeacons are implemented using Bluetooth Low Energy advertisements. As such, a manipulated iBeacon advertisement can be issued with Bluetooth 4 compatible hardware. There are fields within this iBeacon advertisement that can be modified without adversely affecting the transmission of the iBeacon, and as such, provides an opportunity for covert messages. Despite reliability concerns, the technology is continuing to be adopted, and as such, the opportunity for using covert channels over iBeacon is growing as well.

Keywords

Covert channels, iBeacons, Bluetooth Low Energy, Proximity dependent

1. INTRODUCTION

For most concepts in computing, as technology evolves to take advantage of hardware and algorithmic advances, the concept also is adapted to keep pace; however, the core principles of the concept remain. Consider, for example, the concept of outputting information through a computer monitor. Perhaps, originally the monitor displayed information via a monochrome screen, but as technology has advanced, the concept has adapted to utilize color, higher resolutions, or even 3-dimensional illusions; yet through these adaptations, the core principle of displaying information to the user remained.

This idea can apply to the concept of covert channels as well. Butler Lampson first coined the term,"covert channels" in 1973, defining them as "those not intended for information transfer at all..." [1]. Despite the fact that this was written more than 15 years before the invention of the World Wide Web, the original idea has been preserved through the various adaptations and implementations of covert channels for newer technologies. As covert channels have matured and been researched, different characteristics can be defined, regardless of whether the channel was created thirty years ago or yesterday. Some of the important characteristics to understand when discussing a covert channel include mechanism for hiding data, type, throughput, robustness, detection, and prevention [2].

It can be argued that covert channels can actually be created over almost any protocol, examples include TCP/IP, ICMP, ptunnel, DNS, and so on [3]. With some exploration and creativity, opportunities for covert channels can also be unearthed in the many new emerging technologies of mobile and "cloud computing". The aim of this paper is to delve into and describe an opportunity for a covert channel using Apple iBeacons, which is a Bluetooth Low Energy based, proximity dependent technology.

2. UTILITY OF PROXIMITY DEPENDENT COVERT CHANNELS

Most covert channels have been created over networked technologies. Whether the goal is remote command-and-control, exfiltration of data, or other operations, it is likely to be beneficial to the attacker to be able to hide that communication within normal network traffic. However, there are comprehensible circumstances where the network is not a viable medium for covert communication[1]:

- A non-networked device
- A device on a local network with no gateway
- A device on a firewalled network, such that all outbound traffic is denied save for some type that an attacker cannot utilize
- A device on a network under scrutiny for anomalies, such that risk of detection of the attacker's channel is significantly increased

[1]While irrelevant to the focus of this paper, it is interesting to consider how a non-networked or similar device is compromised. It is assumed that in these cases, a malicious insider can gain physical access to the device, or a legitimate user can be phished into compromising the system.

In such instances, it may be useful to utilize a covert channel which relies not on a network medium but on some other method of transmission, most likely between devices located within a close physical proximity between each other. Such technologies may include Bluetooth, near field communication (NFC), wireless networks, sounds, lights (visible or infrared), or similar physical channels. The experiments presented in this paper utilize Apple iBeacons, which rely on Bluetooth technology.

The receiver in a proximity dependent covert channel must also be within the attacker's control. This device can either be the destination for the covert message, or forward it along via legitimate or other covert channels, given it is networked. The following figure shows the simplicity of the concept of a proximity dependent covert channel:

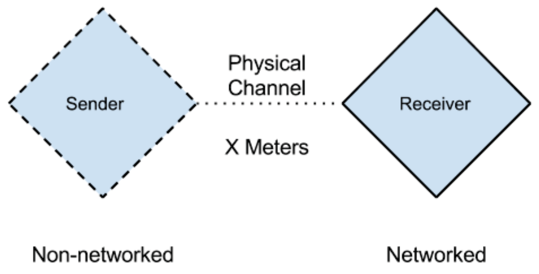

Figure 1: Model - proximity dependent covert channel

To give a real world scenario, let there be a supermarket *XYZ*. *XYZ* processes credit cards during checkout, and to remain PCI-compliant, also segments and firewalls its network such that the outbound traffic from Point-of-Sale (POS) machines can only travel on the secure PCI network. *XYZ* also provides open guest wireless Internet for its customers. *XYZ* has also begun an initiative with its mobile app and in-store iBeacons to help customers find products faster. During off hours, a rogue insider installs credit card sniffing malware onto a POS machine. This rogue insider also slips a nondescript Bluetooth USB dongle into the machine. The following day, as numerous credit card transactions occur, the POS sends iBeacons, disguised as ordinary store iBeacons. Really, these iBeacons covertly transmit credit card numbers. An associate of the rogue insider can then sit in the cafe with his laptop, receive the covert iBeacons and save them, or immediately repackage them and send them out the unfiltered guest wireless, completely undetected.

3. TECHNICAL SPECIFICATIONS OF IBEACON PROTOCOL

The purpose of iBeacons, as described by Apple's iBeacon documentation[4]:

"Introduced in iOS 7, iBeacon is an exciting technology enabling new location awareness possibilities for apps. Leveraging Bluetooth Low Energy (BLE), a device with iBeacon technology can be used to establish a region around an object. This allows an iOS device to determine when it has entered or left the region, along with an estimation of proximity to a beacon"

Apple frameworks automatically handle the low-level specifics of Bluetooth Low Energy and the iBeacon *Protocol*, allowing the developers to incorporate iBeacon functionality simply by configuring a UUID, an optional Major Field, and an optional Minor Field.

Apple's guidance for using these iBeacon fields[4]:

UUID *(16 bytes):* Application developers should define a UUID specific to their app and deployment use case.

Major *(2 bytes):* Further specifies a specific iBeacon and use case. For example, this could define a sub-region within a larger region defined by the UUID.

Minor *(2 bytes):* Allows further subdivision of region or use case, specified by the application developer.

Beyond these 3 fields, developers are given no other means of making modifications to the iBeacon advertisement packet. However, a core feature of iBeacon is the ability to determine an approximation of distance between iBeacon transmitter and the iOS receiver. This is accomplished through the comparison of RSSI (Received Signal Strength Indicator) and *measured transmit power*. The beacon transmits the advertisement packet with a measured transmit power value attached[5]. The receiver then uses both the included measured transmit power and the RSSI of the collected advertisement to determine a distance. Note that this measured transmit power is not set by the developer, but by the device sending the beacon.

However, just like other protocols (e.g. HTTP), there are other pieces to an iBeacon packet that are not explicitly set by the developer/application. There is, unfortunately, no iBeacon public specification. By using packet captures and the Bluetooth Core Spec[6], enough data can be gathered regarding the additional parts of the packet, at least for the purposes of manipulating them into forming a covert channel.[2]

The packet capture partitions an iBeacon advertisement into the following (slightly modified for readability):

```
Parameter Length: 42 (0x2A)
LE Advertising Report
NumReports: 0X01
EventType: Nonconnectable unidirectional advertising
AddressType: Random Device Address
PeerAddress: 0D:EF:97:32:B8:A5
LengthData: 0X1E
Flags: 0x06
Manufacturer Specific Data: –
Data: 02 01 06 1A FF 4C 00 02 15 F1 EB BC 09 A3 13 7F
CD 81 DF 67 C7 79 76 38 88 18 8C 02 43 C0
RSSI: -80 dBm
```

[2]**Open source tool used for sending iBeacons using iOS APIs:** https://github.com/Intermark/Buoy

For the scope of this paper, the *Data* field will be what is analyzed. Upon sending the above iBeacon, the following is known (and configured in code):

UUID: F1 EB BC 09 A3 13 7F CD 81 DF 67 C7 79 76 38 88
Major: 18 8C
Minor: 02 43

Applying this knowledge to the *Data* field, it becomes:

Data: 02 01 06 1A FF 4C 00 02 15 [UUID] [Major] [Minor] C0

From this *Data* field, it is also known that **4C 00** is the Bluetooth company identifier for Apple[7].

As will be discussed later in the paper, the remaining octets prior to the UUID appear to be part of the iBeacon Prefix, BLE Flags, or BLE advertisement packet requirements; in summary, these are what identifies an iBeacon as an iBeacon (manipulations to these octets causes the transmission to no longer be detected by an iBeacon listener).

It was also determined that the last octet in the data field is the transmitted power, as set by the sender. Note that this is not the *RSSI*, which is the actual received signal strength.

With this information, the summary of the iBeacon *Data* field is (**size in bytes within parenthesis**):

Data:
Prefix/Flags (**5**)
Company ID (**2**)
Prefix/Flags (**2**)
UUID (**16**)
Major (**2**)
Minor (**2**)
Transmitted Power (**1**)

To summarize for the purposes of describing this covert channel, it is important to understand an *iBeacon* is:

- designed to provide additional location-based information to an iOS device
- a unidirectional Bluetooth Low Energy advertisement (i.e broadcast)[3]
- without a true data payload–the way it conveys information is through identifiers and signal strength (however, it is important to note the various components of the *Data* field)

[3]For in-depth information about Bluetooth Low Energy, refer to the core specification document[6]

4. REQUIREMENTS AND CHARACTERISTICS OF IBEACON COVERT CHANNELS

An iBeacon covert channel can be simplified by the following 2 requirements:

- It must be capable of transmitting a covert message using iBeacon packet(s) as a vehicle
- Upon reception by a legitimate iBeacon receiver (i.e. an Apple iOS device), the packet must be decipherable and interpreted as an iBeacon.

These requirements, however, sit on top of the traditional requirements and characteristics of covert channels:[4]

- **Mechanism for hiding data**: The hidden message must be somehow encoded within the iBeacon advertisement packet. The idea is that not only is there a secret message, but to outside observers, it appears that no data-exchange is even taking place. Because iBeacons are not designed to transmit data payloads, they inherently do not appear to be transmitting extra data.
- **Type**: iBeacon advertisements could potentially be used for either *storage based* or *timing based* types of covert channels. However, the lack of reliability of delivery of advertisements makes timing channel opportunities much more challenging.
- **Throughput**: Throughput of an iBeacon covert channel is dependent on the amount of data sent per advertisement, the interval at which advertisements are broadcasted (which, based upon the BLE spec requirements, is between 20ms and 10.24s[8]), and any necessary retransmissions, due to the volatile nature of BLE.
- **Robustness**: A proximity based covert channel requires a certain distance be maintained between sender and receiver. For BLE-based iBeacons, other physical factors could also influence the successful delivery of packets, such as walls, humans, or other objects impeding the path of the signal.
- **Detection**: Because iBeacons are broadcast on physical radio waves, packets are not traversing networks or the detection tools that would be placed on them, such as intrusion dectection systems, firewalls, or SIEMS. As such, the most obvious method of detection would be Bluetooth sniffers that analyze BLE packets. Given the existence of such a device, it then must detect that a given iBeacon is anomalous.
- **Prevention**: Prevention is difficult without sacrificing the availability of Bluetooth or emerging technologies they provide, such as iBeacons.

The iBeacon covert channel discussion in the next section builds upon these requirements and characteristics.

[4]This list of characteristics is based upon the research completed by Johnson et al.[2]

5. IBEACON COVERT CHANNEL

An analysis of each of the iBeacon *Data* fields can show where opportunities for covert channels exist.

5.1 Prefix/Flags

Modifications to the elements of either of the *Prefix/Flags* fields results in the Bluetooth advertisement no longer being recognized as an iBeacon by an iOS device, which breaks one of the core requirements for this covert channel. As such, none of these bytes provide opportunity for an iBeacon covert channel.[5]

5.2 Company ID

Apple's company identifier for Bluetooth transmissions is **0x004C**, as registered with the Bluetooth SIG[7]. Note that as this is transmitted as a BLE advertising packet, it follows the *Little Endian* format, with the least significant bit first.[6] As shown in the example in the previous section, this company identifier is physically sent as the bytes **4C 00**. It was discovered that modifying the **00** byte to some other value than *00* resulted in the advertisement continuing to be recognized as an iBeacon. Modifying the **4C** byte causes the iBeacon receiver to ignore the packet, which indicates that some sort of validation of the company identifier field does occur. However, it appears to only check the first byte. Perhaps this is an oversight on Apple's implementation, or perhaps it is for optimization. Regardless this 2nd company identifier byte can be modified and included in an iBeacon transmission without adversely affecting the successful delivery of the iBeacon. The location of this byte, in relation to the earlier packet example (where *XX* indicates the modifiable company identifier byte):

Data: 02 01 06 1A FF 4C **XX** 02 15 F1 EB BC 09 A3 13 7F CD 81 DF 67 C7 79 76 38 88 18 8C 02 43 C0

5.3 UUID, Major, Minor

Modifications to the UUID, Major, or Minor bits do not affect the validity of an iBeacon transmission. In fact, these fields are promoted to developers as the way iBeacons work. Using these three fields as a channel would yield 20 bytes of data per transmission, 16 for the UUID, 2 for the Major, and 2 for the Minor. At most however, this would be considered an obscured channel, simply because of the unlikelihood of the channel being monitored. Given a suitable device, such as a BLE packet sniffer, detection of this anomalous traffic is a certainty, as the UUID, Major, and Minor bytes would not match any of the expected *production* iBeacons. As such, modifying any of these fields is out of scope for a true covert channel.

5.4 Transmitted Power

Typically, the transmitted power represents a measurement by the Bluetooth device manufacturer at one meter away[5]. This calibrated transmitted power is then sent with the transmission of the advertisement. The receiver can then compare the calibrated transmitted power with the RSSI (Received Signal Strength Indicator). The general concept for determining distance is if two advertisements with identical calibrated transmitted power values are received, the packet with a stronger RSSI has a closer sender.

Apple API provides developers 4 descriptors when calculating range from an iBeacon: *immediate, far, near, unknown*[4]. These descriptors are not very precise, and for valid reasons as RSSI has been shown to not be a very reliable method for determining the distance between objects[10].

The transmitted power byte can be modified to any valid hex character without breaking the iBeacon protocol. If the receiver cannot process the difference between calibrated transferred power, it will return the unknown descriptor; otherwise it will compute the range as immediate, far or near. All of these result in a valid iBeacon. As such, this byte provides a clear opportunity of another option for a covert channel. The location of this byte, in relation to the earlier packet example (where *YY* indicates the modifiable transmitted power byte):

Data: 02 01 06 1A FF 4C XX 02 15 F1 EB BC 09 A3 13 7F CD 81 DF 67 C7 79 76 38 88 18 8C 02 43 **YY**

5.5 Theoretical Covert Channel Throughput

In a theoretical context, the following assumptions are made:

- Every iBeacon sent is always received
- iBeacons are able to be sent every 20ms (the shortest technically possible interval for BLE advertisements)

This leads to a throughput of 2 bytes (one in the company identifier and one in the transmitted power) every 20ms.

In more common terms, the maximum possible **throughput** is: **100 bytes per second**.

5.6 Practical Covert Channel Throughput

The actual throughput of an iBeacon covert channel is difficult to estimate. Because of the volatility of BLE advertisements and objects in the physical environment, it is not likely that all iBeacons transmitted by the sender will be received. Further, because of expected iBeacon loss, the covert channel implementation will probably have to sacrifice a byte as some kind of *sequence number*, similar to the sequence number used in TCP/IP. This drops the throughput down to 1 byte per transmission. In the proof of concept, it will be shown that the actual throughput is more on the scale of 1 byte per 3 seconds.

6. PROOF OF CONCEPT

6.1 Tools

Sender
Kali Linux machine equipped with the BlueZ stack.[6]
Satechi USB 4.0 Bluetooth Adapter.

[5]This does not conclusively determine that *Prefix/Flags* fields do not contain opportunity for BLE covert channels, simply not for an iBeacon specific channel. For more information on the BLE advertising packet, see Bluetooth Core Spec, Volume 3, Part C, Section 11 and Section 18.[9]

[6]Official Linux Bluetooth protocol stack. www.bluez.org

Receiver
Macbook Pro (late 2013) equipped with internal Broadcom Bluetooth device and Bluetooth sniffing software.

Normal iBeacon Advertisement Listener
iPad Air equipped with the Locate Beacon app[7]

6.2 Methods

Both the company identification byte and the power byte are utilized in the following proof of concept. However, to compensate for the expected volatility of Bluetooth Low Energy advertisements, the power byte is used only as a sequence number, not as a data payload.

Sending iBeacons

The following command will instruct the Bluetooth interface to configure the advertising payload (length of payload specified as *1e*)[8]:

```
hcitool -i hci0 cmd 0x08 0x0008 1e $payload
```

where an example **$payload** is (XX representing covert channel data, YY representing a sequence number):

```
02 01 06 1A FF 4C XX 02 15 F1 EB BC 09 A3 13 7F CD
81 DF 67 C7 79 76 38 88 18 8C 02 43 YY
```

To instruct the Bluetooth interface to begin advertising the configured packet:

```
hciconfig hci0 leadv
```

Changing the Advertising Interval

The default advertising interval is 1.28 seconds[9]. The following commands were issued to change the interval to 100ms (derived using Bluetooth Core Spec, Volume 2, section 7.8.5) and then begin advertising at that rate:

```
hcitool -i hci0 cmd 0x08 0x0006 A0 00 A0 00 03 00 00 00 00
00 00 00 00 07 00
hcitool -i hci0 cmd 0x08 0x000a 01
```

Receiving iBeacon Advertisements

For a real-world deployment of this covert channel, the tool *hcidump* will probably be the most useful for scripts and parsing iBeacons. An example command for sniffing Bluetooth packets and outputing hex:

```
hcidump -i hci0 -x
```

For the sake of this proof of concept and human-readability, a GUI OSX Bluetooth packet sniffer was used.

Distinguishing Covert Channel iBeacons

An important attribute of a covert channel is the method the receiver uses to tell the difference between legitimate traffic and covert traffic. In this iBeacon covert channel, it is quite easy to distinguish the covert packets; if the company identifier is set to any other value than *4C 00*, it is known to be an iBeacon covert channel. It is particularly important for those using this channel to recognize that *00* is not a valid value for the covert channel.

Other Implementation Considerations

There are other details that may be desirable to consider while creating this covert channel:

- Length of message, including how to handle sequence number (e.g. should *FF* roll over to *00*?)
- Duration of time to advertise each payload
- How to specify the end of the message
- Retransmitting missed advertisements

However, many of these are dependent on use-case, payload type, and other details that are beyond the scope of this proof of concept.

7. RESULTS AND DISCUSSIONS

7.1 Sending a Message

Covert message transmission was successful for the message *123456*. A 3 second duration of advertising was chosen for each byte of the message, which proved necessary, as in one of the experiments, only one advertisement was received containing the *12* byte of the message. For the sake of simplicity, *00* was chosen as the first sequence number and then incremented for each following value advertised (sequence number - value, *00 - 12, 01 - 34, 02 - 56*). It should be noted that this choice of sequence number results in iBeacons with an *unknown* distance.

7.2 Advertising Speed

With the default advertising interval, an experiment was conducted simply measuring how many packets were received by the receiver. In 1475 seconds, only 105 advertisements were received, which corresponds to approximately 1 received advertisement every 14 seconds.

After the advertising interval was modified to send every 100ms, the experiment was run again. This time, after 401 seconds, 629 advertisements had already been received. This corresponds to 1 advertisement received every .64 seconds.

For these experiments, the sender and receiver were placed approximately one meter apart. Different results would be expected with different hardware or environments. However,

[7]Created by Radius Networks
[8]The commands sent to the interface: 0x08 - OGF Code for LE Controller Commands; and 0x0008 - LE Set Advertising Data, see Bluetooth Core Spec, Volume 2, section 7.8 and 7.8.7[9]

the concept of faster advertising intervals correlating with more quickly received packets should still apply.

7.3 Importance of sequence numbers

Because there is a duration of time where the same value is advertised over and over, it is crucial to specify when an advertisement contains a new value. This is accomplished through the use of a sequence number in the transmitted power field. If the sequence number has changed from the previous packet, the data is new, otherwise, it should not be appended to the secret message. Using a sequence number also assists with keeping the packets ordered, or determining when an advertisement is completely unreceived (e.g. when the sequence numbers received go from 04 to 06, skipping 05).

7.4 Reliability concerns

As has been apparent throughout much of this paper, advertisement loss is very prevalent with iBeacons, or at least the implementation used for this proof of concept. It is unknown whether the problem resided with the sender not sending as fast as it was configured to (due to limited resources/hardware), the receiver dropping packets (with similar resource or driver limitations), elements in the environment, or a combination of all of the above. However, the message tests and the speed tests make it very apparent that at some point in the system, advertisements were lost. While this issue is mitigated to a certain extent by the sequence numbers solution, this cuts into potential throughput of the channel.

8. CHALLENGES AND CONCLUDING REMARKS

8.1 Challenges of Covert Channels in Emerging Technologies

Bluetooth Low Energy is a relatively new field, and with technology advancements in hardware, is susceptible to volatility. As an example, mere software updates to the operating system of one of the experiment machines caused the Bluetooth interface to work differently. Similarly, iBeacon, as a proprietary advertisement format on top of BLE, could change the way it operates. These technologies might not even exist in years to come. All of these caveats factor into the development and research into covert channels over these mediums. Furthermore, this covert channel, as proposed, relies upon the fact that the company identifier field is not verified in full. In the event this is rectified, covert channels would have to be created using other techniques (perhaps timing channels monitoring the transmitted power field, or similar).

8.2 Concluding Remarks

After delving into the packet structure and configuration of Bluetooth Low Energy advertisements, it is clear that opportunities are present for covert communication over iBeacon advertisements. It is also clear that there are many challenges surrounding using this technology, especially with overcoming reliability issues; however, these are the same challenges that have to be solved for the normal usage of this technology. As the adoption of iBeacon technology grows, the opportunities for this covert channel will expand as well. As further research is conducted in emerging technologies utilizing proximity dependent channels, such as the discussed iBeacon, it will be important to consider the possibility of covert channels; traditional network security equipment might not cover this new communication space.

9. REFERENCES

[1] B. W. Lampson, "A note on the confinement problem," *Commun. ACM*, vol. 16, pp. 613–615, Oct. 1973.

[2] D. Johnson, B. Yuan, P. Lutz, and E. Brown, ""covert channels in the http network protocol: Channel characterization and detecting man-in-the-middle attacks","
`http://scholarworks.rit.edu/other/781/`.

[3] E. Couture, "Covert channels," *SANS Institute*.

[4] Apple, Inc., *Getting Started with iBeacon*, 1.0 ed., June 2014. `https://developer.apple.com/ibeacon/Getting-Started-with-iBeacon.pdf`.

[5] T. Andersson, "Bluetooth low energy and smartphones for proximity-based automatic door locks," 2014.

[6] Bluetooth SIG, *Bluetooth Core Specification 4.1*.
`https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=282159`.

[7] Bluetooth SIG, *Company Identifiers*.
`https://www.bluetooth.org/en-us/specification/assigned-numbers/company-identifiers`.

[8] J. Liu and C. Cheng, "Energy analysis of neighbor discovery in bluetooth low energy networks," *Technical Report, Nokia Research Center/Radio Systems Lab.*

[9] Bluetooth SIG, *Bluetooth Core Specification 4.0*.
`https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=229737`.

[10] A. T. Parameswaran, M. I. Husain, and S. Upadhyaya, "Is rssi a reliable parameter in sensor localization algorithms - an experimental study," *Field Failure Data Analysis Workshop*, 2009.