

Aditya Gupta (@adi1391)
presents:

Hacking Bluetooth Low Energy for Internet of Things

About the Workshop

AGENDA?

- + Internals of BLE
- + BLE Security
- + Interacting with BLE devices
- + Sniffing BLE traffic
- + Controlling a device
- + BLE exploitation scripting

WHO IS THIS GUY?

- + Aditya Gupta (@[adi1391](https://twitter.com/adi1391))
- + Founder of Attify - specialized IoT security firm
- + Creator of Attify-Store (ihackiot.com)
- + "Offensive IoT Exploitation" training - iotsecuritytraining.com
- + Author of Mobile and IoT security books
- + Spoken at various cons earlier -BH, Defcon, OWASP AppSec, phdays and more.

Since this is an extremely short workshop with limited time, we will focus on practical attack scenarios and keep theoretical information to a minimum. The entire workshop is **focused around exploiting the BLE devices** which we have. Due to limited time, we won't be able to go into topics such as BLE authentication, pairing, encryption details (but will cover them briefly). Please feel free to talk to us post-workshop if interested in these topics.

Hello world!

I AM ADITYA GUPTA

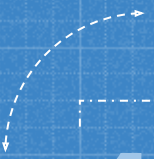
What do you know about
BLE Security?

Have you hacked BLE
devices earlier?

This is going to be a
beginner-friendly class!

If you are expecting
advanced topics, sorry we
won't have time to cover
that.



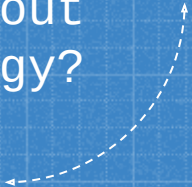


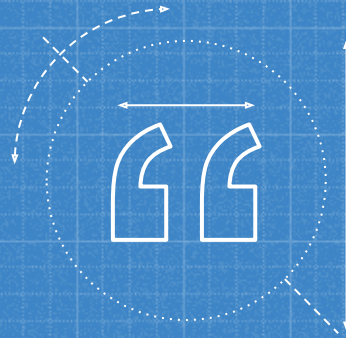
1

WHY BLE?



Why do we even care about
Bluetooth Low Energy?





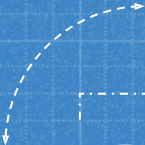
BLE is one of the most common communication medium used in IoT devices - especially Smart homes, Fintech and medical devices.

What is BLE?

- Bluetooth Low Energy, Bluetooth Smart, sub(Bluetooth 4.0 spec)
- Really, really different from the traditional Bluetooth
- Light weight, low power and resource consumption
- Battery can last for months to years
- Works on 2.4 GHz (ISM band) with a max range of around ~100m

Some devices using BLE?

- Smart home
- Smart mirrors
- Smart Coffee Machines
- Smart Plugs
- Smart enterprise devices
- Medical devices
- Payment systems
- And many more



2

ATTACK SURFACE MAPPING

What are the various entry
points?

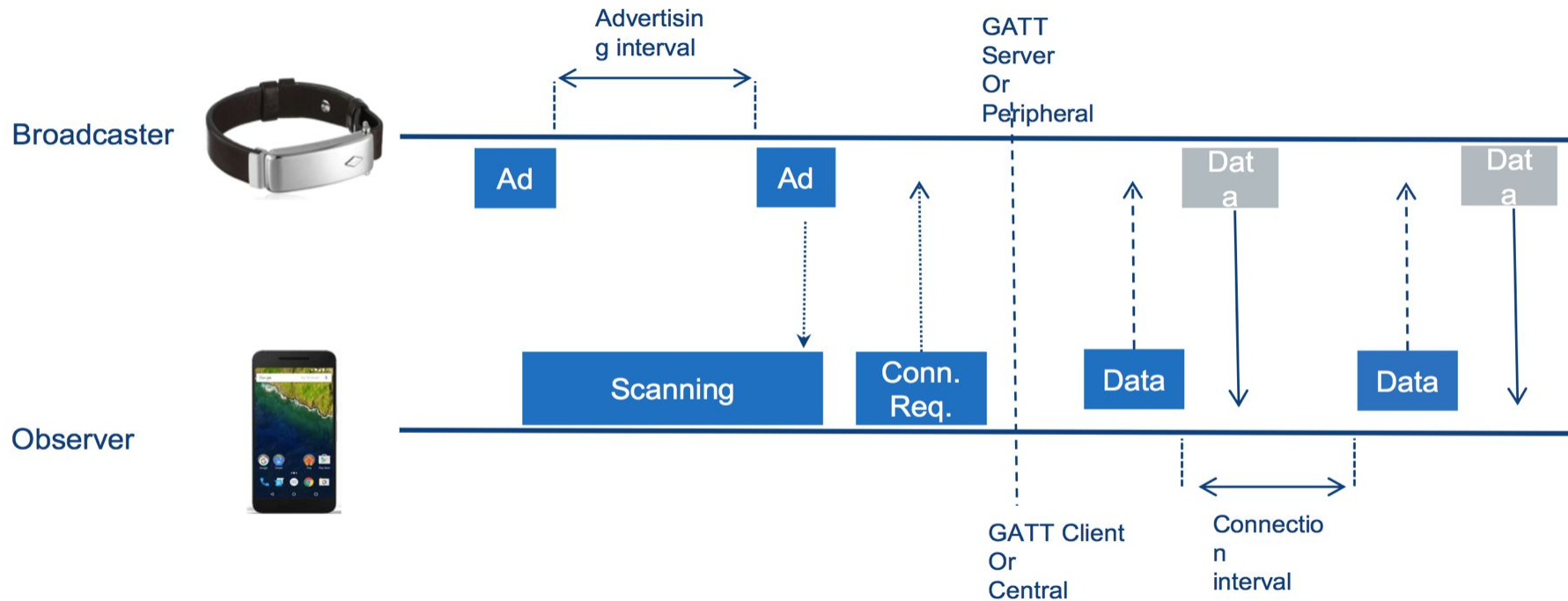
MAPPING BLE ATTACK SURFACE

- Understand the device and it's functionality
- See how it communicates
- What are the various data components that it exchanges
- Sniff, Replay, MITM etc.

MAPPING BLE ATTACK SURFACE

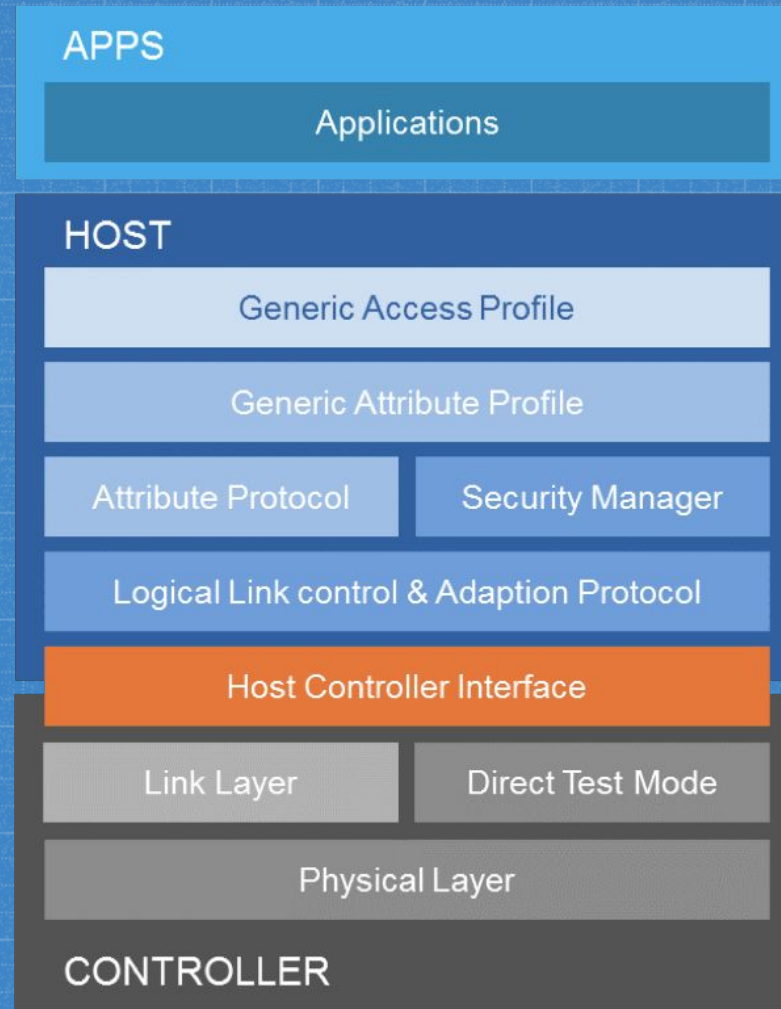
- In order to map the attack surface, we need more info
- How do BLE devices pair?
- How do BLE devices communicate?
- What does the architecture look like?

HOW DO BLE DEVICES PAIR?



(Image source - When Encryption is Not Enough by Sumanth Naropanth, Chandra Prakash Gopalaiah & Kavya Racharla)

BLE STACK

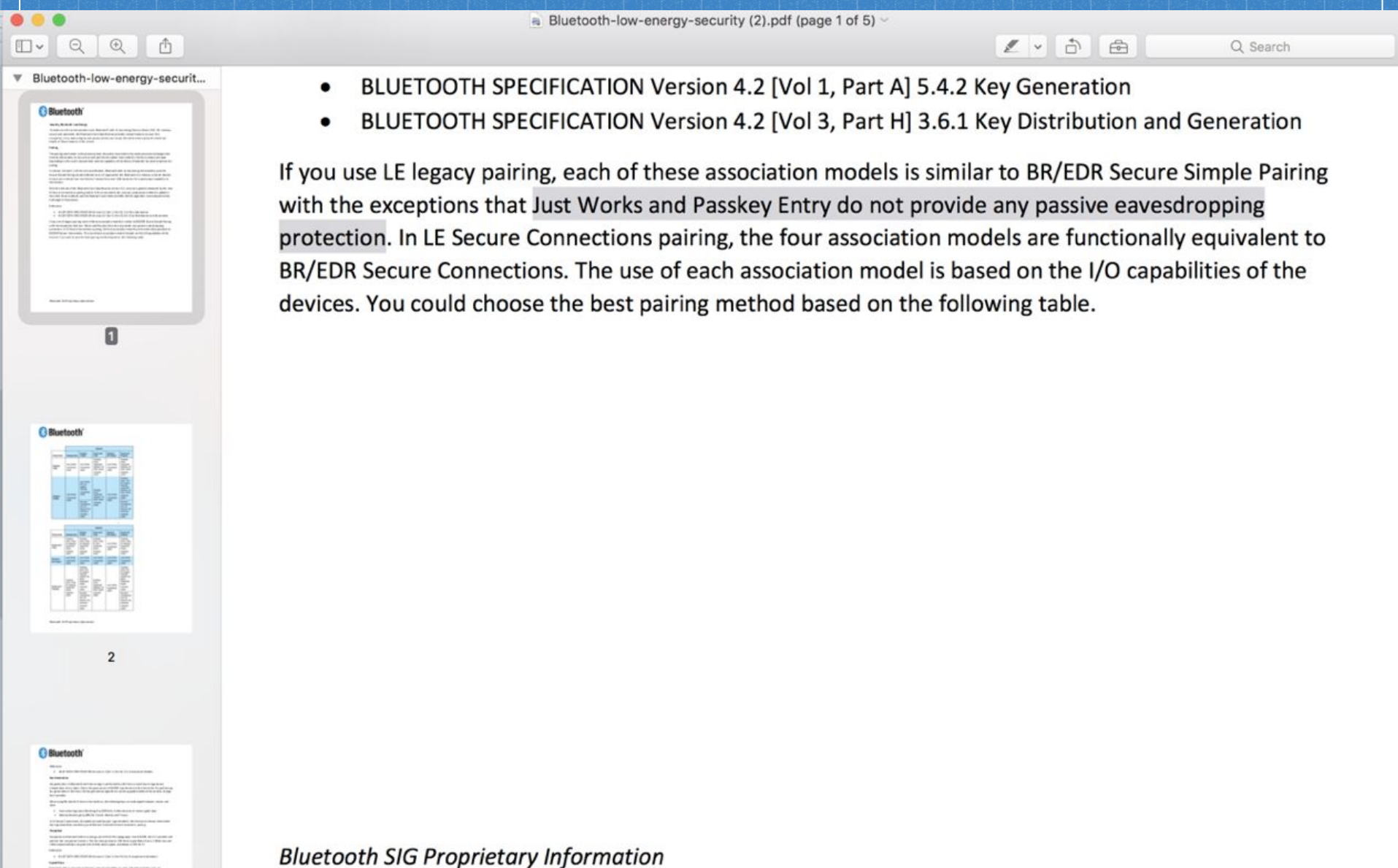


Source : <https://www.bluetooth.com/specifications/bluetooth-core-specification>

PAIRING MODES

- Just Works (000000)
- Numeric Comparison (Yes/No)
- Passkey (6-digit)
- Out-of-Band

As per the official docs.

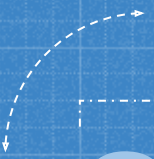


CRACKLE

```
/home/oit/lab/BLE/crackle-0.1/crackle-sample [oit@ubuntu] [12:21]  
> crackle -i ltk_exchange.pcap -o decrypted.pcap  
TK found: 000000  
ding ding ding, using a TK of 0! Just Cracks(tm)  
Warning: packet is too short to be encrypted (1), skipping  
LTK found: 7f62c053f104a5bbe68b1d896a2ed49c  
Done, processed 712 total packets, decrypted 3
```

CRACKLE

```
/home/oit/lab/BLE/crackle-0.1/crackle-sample [oit@ubuntu] [12:23]  
> crackle -i encrypted_known_ltk.pcap -o new_decrypted.pcap -l 7f62c053f104a5bbe68b1d896a2ed49c  
Warning: packet is too short to be encrypted (1), skipping  
Warning: packet is too short to be encrypted (2), skipping  
Warning: could not decrypt packet! Copying as is..  
Warning: could not decrypt packet! Copying as is..  
Warning: could not decrypt packet! Copying as is..  
Warning: invalid packet (length too long), skipping  
Done, processed 297 total packets, decrypted 7
```

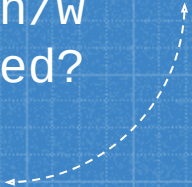


3

LAB FOR BLE EXPLOITATION



What are the tools and h/w
you need to get started?



SETTING UP YOUR BLE LAB - S/W

- Wireshark
- Blue Hydra
- Ubertooth-utils
- hcitool
- Gatttool
- BTLEJuice
- Gattacker
- Usual utilities - hexdump, bless, strings etc.

SETTING UP YOUR BLE LAB - H/W

▪ **BLE DONGLE**

- For sending BLE packets
- Quick and handy to interact with BLE devices

▪ **Ubertooth One**

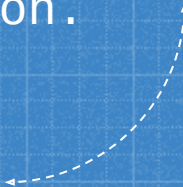
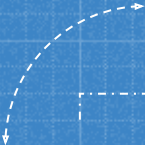
- Works well for sniffing
- Both for active and passive analysis

▪ **BLE SNIFFER (ADA)**

- Inexpensive
- Works (kinda) well
- Integration with Wireshark
- Cons - Windows

▪ **RPi 3**

- Comes with a BLE interface
- Provides portability for attacks
- Automation?



4

GETTING STARTED

Recon.

First step - RECONNAISSANCE

- Understand the device that you're going to attack
- Find out as much information as possible about the device
- BD_ADDR, RSSI, Device name etc.
- How are the devices communicating?

Quick Exercise

- Understand the device that you're going to attack
- Find out as much information as possible about the device
- BD_ADDR, RSSI, Device name etc.
- How are the devices communicating?

HCITOOl

HCITOOl(1)

Linux System Administration

HCITOOl(1)

NAME

`hcitool` - configure Bluetooth connections

SYNOPSIS

```
hcitool [-h]  
hcitool [-i <hciX>] [command [command parameters]]
```

DESCRIPTION

`hcitool` is used to configure Bluetooth connections and send some special command to Bluetooth devices. If no **command** is given, or if the option **-h** is used, `hcitool` prints some usage information and exits.

OPTIONS

-h Gives a list of possible commands

-i <hciX>
The command is applied to device `hciX`, which must be the name of an installed Bluetooth device. If not specified, the command will be sent to the first available Bluetooth device.

COMMANDS

dev Display local devices

inq Inquire remote devices. For each discovered device, Bluetooth device address, clock offset and class are printed.

scan Inquire remote devices. For each discovered device, device name are printed.

name <bdaddr>
Print device name of remote device with Bluetooth address `bdaddr`.

info <bdaddr>

Manual page hcitool(1) line 1 (press h for help or q to quit)

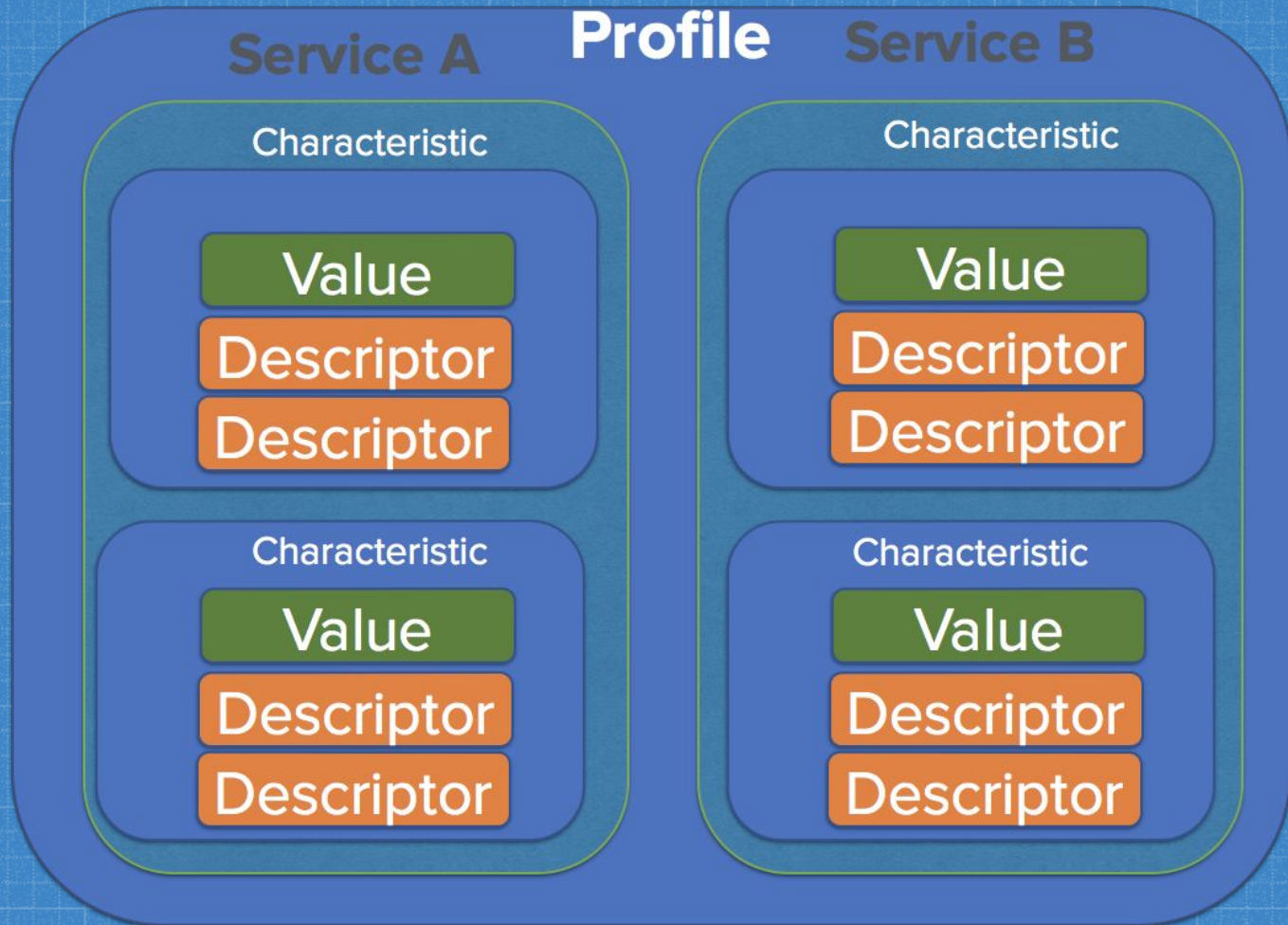
Understanding GATT and GAP

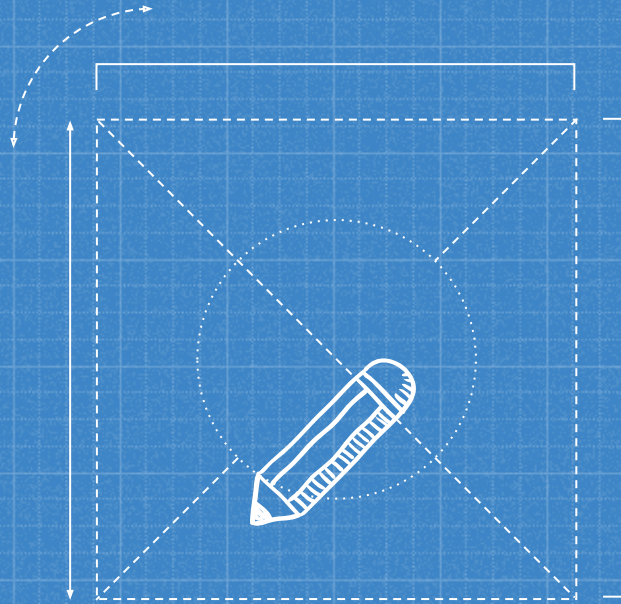
- **Generic Attribute Profile (GAP)** - responsible for discovery and basic setup
- **Attribute Protocol (ATT)** -
 - defines the client/server protocol for data exchange
 - Grouped together into meaningful services using the GATT
- Responsible for the exchange of all user data and profile information in a BLE connection
- GATT-based profiles as per the Bluetooth SIG ensuring device interoperability

Services and Characteristics

- Services can contain multiple characteristics
- Characteristics, services and descriptions are combined called Attributes
- Identified by associated UUID Depending on the properties, other devices could
 - Read
 - Write
 - Subscribe to notifications
- Services can be accessible with or without authentication

PROFILES





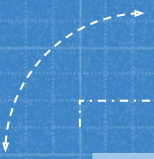
HANDS-ON

Use the [DEVICE 1] given to you and **connect**
to it using Gatttool

Explore the **various services and**
characteristics on that device

What we just did?

- `hcitool lescan` (or `bluehydra scan`)
- Identified the address
- `gatttool -I -b [addr]`
- `primary`
- `characteristics`
- `char-read-hnd hnd-value`
- Look it up on the Bluetooth SIG website

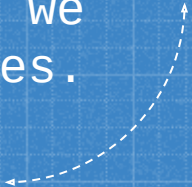


5

EXPLOITING BLE



Now is the time when we
start attacking devices.



TYPES OF ATTACKS IN BLE

Sniffing

Sniffing a given device to look at it's entire data exchange that is happening - does it uses clear text data transmission? What kind of info is revealed?

MITM and Replay

Can you replay a given communication packet that you've captured? What are the consequences of the replay attack? Opening lock? Authentication?

Modifying attributes

Are you able to connect to the device? If yes, what if you can modify the values of various handles and make the device behaves otherwise.

Jamming

Can we jam the signal? If yes, what are the consequences of the signal being jammed? Think from a real-world attackers perspective.

Getting started with [Device 1]

- Connect to the Beacon using Gatttool
- Read all the various characteristics and handlers
- See which ones stand out (not defined in the Bluetooth SIG)
- Modify the values of that handler
- Poke until it hurts!
- What do you see/hear?

Getting started with [Device 1]

```
[FF:FF:E0:00:36:EB][LE]> characteristics
handle: 0x0002, char properties: 0x12, char value handle: 0x0003, uuid: 00002a00-0000-1000-8000-00805f9b34fb
handle: 0x0004, char properties: 0x02, char value handle: 0x0005, uuid: 00002a01-0000-1000-8000-00805f9b34fb
handle: 0x0007, char properties: 0x12, char value handle: 0x0008, uuid: 00002a19-0000-1000-8000-00805f9b34fb
handle: 0x000a, char properties: 0x1c, char value handle: 0x000b, uuid: 00002a06-0000-1000-8000-00805f9b34fb
handle: 0x000d, char properties: 0x12, char value handle: 0x000e, uuid: 0000ffe1-0000-1000-8000-00805f9b34fb
Notification handle = 0x000e value: 01
```

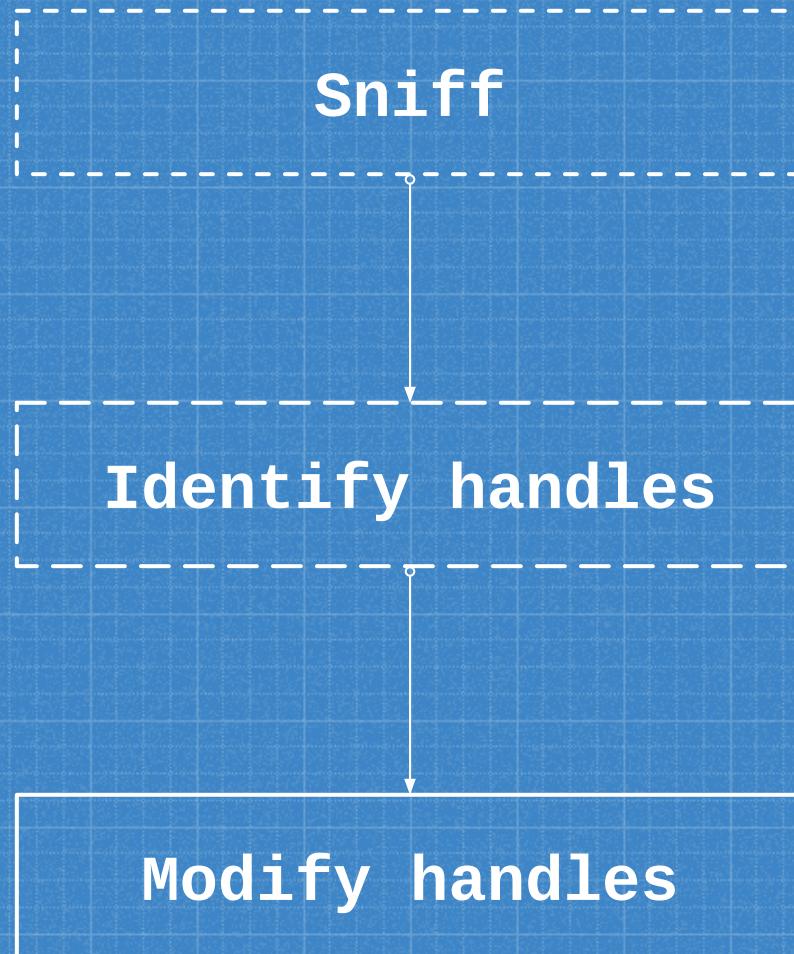
Getting started with [Device 1]

- 0x2a00 corresponds to device name
- 0x2a01 corresponds to Appearance
- 0x2a06 corresponds to **Alert level**
- 0x2a19 corresponds to **Battery level**

How do you know what to change?

- Yes we told you?
- But in real-life?
- Sniffing is the answer?
- We will all now sniff packets - let's hope the hands-on exercise works - if not sample packet capture!

BLE hacking process



Ubertooth One

- Developed by Michael Ossmann
- CC2400 + NXP LPC1756 + USB 2.0
- Identify clear text data transmission
- Identify which handlers are being written and read
- Manually modify those handlers

Ubertooth One

- Use `-f` for the follow mode
- `sudo ubertooth-btle -f -t [address]
-c file.pcap`
- Analyze it in Wireshark
- use filter - `bt12cap.cid==0x04`
- Do you see any useful data?

Real-time sniffing

- Can also create a pipe and pass the data from ubertooth sniff to pipe
- Use the pipe as capture interface in Wireshark
- `sudo mkfifo /tmp/pipe`
- `sudo ubertooth-btle -f -t [address] -c /tmp/pipe`
- In wireshark - add this as a new interface and then sniff over this

Exploiting a LightBulb

03	C9	00	06	00	0A	03	00	01	01	00	00	24	FF	00	00	00	00
----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----	----



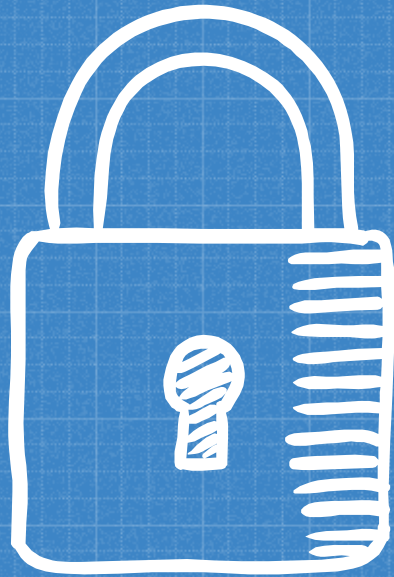
HEADER



**ON/OFF
BIT**



**COLOR
(RGB)**



HACKING A BLE SMART LOCK

Hacking a SmartLock

- Find the Bluetooth address using hcitool for the SmartLock
- Starts advertising once you hit the button
- Install [app-name] on your mobile phone
- Set up [device] to sniff packets for your target (Addr on back of your device)
- Is the **password in clear text**? How is it **unlocking the device**?
- Can you transmit those same packets by yourself (hcitool) or change the handlers (gatttool)?

Let's automate?

- Introduction to PyGatt
- Connect to the adapter
- What action do you want to perform?
- Automate the process (send both the requests that is required to exploit the device)
- `python exploit-lock.py`

Thanks!

ANY QUESTIONS?

You can find me at:

@adi1391

adi@attify.com

CREDITS

Special thanks to all the people who made and released these awesome tools:

- Ubertooh One - Michael Ossmann
 - Blue Hydra - Pwnie Express
 - Gattacker - Slawomir Jasek
 - Manufacturers - for making vulnerable devices
 - Defcon - for selecting this workshop
 - Attify team members - for putting all the effort behind this workshop
 - You - for attending the workshop and registering for it like a ninja
-
- Enjoy the rest of the event. Find us around. Tweet - #defconiot