Having fun with IoT: Reverse Engineering and Hacking of Xiaomi IoT Devices
DEFCON 26 – Dennis Giese
Outline

• Motivation
• Xiaomi Cloud
• Overview of devices
• Reverse Engineering of devices
• Modification of devices
About me

• Researcher at Northeastern University, USA
  – Working with Prof. Guevara Noubir@CCIS

• Grad student at TU Darmstadt, Germany
  – Working with Prof. Matthias Hollick@SEEMOO

• Interests: Reverse engineering of interesting devices
  – IoT, Smart Locks
  – Physical Locks ;)

• [Insert more uninteresting information here]
MOTIVATION
Why reverse IoT?

• (Find and exploit bugs to hack other people)
• De-attach devices from the vendor
• Enhance functionality
  – Add new features
  – Localization (e.g. Sound files)
  – Defeat Geo blocking
• Supporting other researchers
Mon(IoT)or Lab@NEU

https://moniotrlab.ccis.neu.edu/
“Responsible disclosure”?

- Ethical question: “Responsible disclosure”?
  - Conflict:
    - Rootability vs. Device security
    - “Service for the Community” vs. Bug Bounty Program
  - Before DEFCON: contacted Xiaomi security team
How we started

May 2017
Mi Band 2
Vacuum Robot Gen 1

June 2017
Lumi Smart Home Gateway + Sensors

July 2017
Yeelink Lightbulbs (Color+White)
Yeelink LED Strip

Research in cooperation with Daniel Wegemer
How we continued

- Yeelink Desk lamp
- Philips Eyecare Desk lamp
- Xiaomi Wi-Fi router
- Yeelink/Philips Ceiling Lights
- Philips Smart LED Bulb
- Vacuum Robot Gen 2
- Yeelink Bedside Lamp
- Xiaomi (Ninebot) M365
- Lumi Aqara Camera
- Yeelink Smart LED Bulb (v2)
- Smart Power strip
Why Vacuum Robots?

Three Processors

To provide more location stability there are three dedicated processors to track its movements in real-time, calculate the location and determine the

Source: Xiaomi advertisement
THE XIAOMI CLOUD
Xiaomi Cloud

• They claim to have the biggest IoT ecosystem worldwide
  – 85 Million Devices, 800 different models
• Different Vendors, one ecosystem
  – Same communication protocol
  – Different technologies supported
  – Implementation differs from manufacturer
• Software quality very different

Xiaomi Ecosystem

Cloud Protocol (WiFi)

- WiFi
- BLE
- ZigBee

Gateway

HTTPS

Xiaomi Cloud (Mi)

* There could be more connections (e.g. P2P, FDS)
Device to Cloud Communication

• DeviceID
  – Unique per device

• Keys
  – Cloud key (16 byte alpha-numeric)
    • Is used for cloud communication (AES encryption)
    • Static, is not changed by update or provisioning
  – Token (16 byte alpha-numeric)
    • Is used for app communication (AES encryption)
    • Dynamic, is generated at provisioning (connecting to new Wi-Fi)
Cloud protocol

• Data
  – JSON-formatted messages
• Example of “Device registration”
Protocol for Firmware updates

• APP Updates
  – {"method":"miO.ota","params":{"app_url":"http://cdn.cnbj0.fds.api.mi-img.com/miio_fw/upd_lumi.gateway.v3.bin?...","file_md5":"063df95bd5....cf11e","install":"1","proc":"dnld install","mode":"normal"},"id":123}

• MCU/WiFi Updates
  – {"method":"miO.ota","params":{"mcu_url":"http://cdn.cnbj0.fds.api.mi-img.com/miio_fw/mcu_lumi.gateway.v3.bin? ...","install":"1","proc":"dnld install","mode":"normal"},"id":123}

• Subdevice Updates
  – {"crc32":"9460d9f0","image_type":"0101","manu_code":"115F","md5":"e9d62...a74d 8","model":"lumi.plug.v1","size":"186978","url":"http://cdn.cnbj2.fds.api.mi-img.com/lumi-ota/aiot-ota/LM15_SP_mi_V1.3.22___OTA_v22_withCRC.ota"}
Xiaomi Ecosystem

Cloud Protocol (WiFi)

HTTPS

Xiaomi Cloud

WiFi

BLE

ZigBee

Gateway
App to Cloud communication

- Authentication via OAuth
- Layered encryption
  - Outside: HTTPS
  - Inside: AES using a session key
- Message format: JSON RPC
- Device specific functions: provided by Plugins
App to Cloud communication

- **REQ:** api.io.mi.com/home/device_list method:POST params:[]
- **RES:**
  
  ```json
  {"message":"ok","result":{"list":[{"did":"659812bc...zzz","name":"Mi PlugMini","localip":"192.168.99.123","mac":"34:CE:00:AA:BB:CC","ssid":"IoT","bssid"DD:EE","model":"chuangmi.plug.m1","longitude":"-71.0872248","latitude":"42.33794500","adminFlag":1,"shareFlag":0,"permitLevel":16,"isOnline":true,"desc":"Power plug on ","rssi":-47}
  ```
App to Cloud communication

- "longitude":"-71.0872248","latitude":"42.33794500"
Example of Communication relations

- **Miio_client**
  - (local):54322 (tcp)
  - 0.0.0.0:54321 (udp)

- **player**
  - 0.0.0.0:6665

- **RoboController**
- **AppProxy**
- **wifimgr**

- **device.conf**: DID, Key

- ***.fds.api.xiaomi.com**: (https)

- **ott.io.mi.com**: 80 (tcp)
  - 8053 (udp)

- **AES encrypted**

- **Android/iPhone App**

- **IPC**
  - plain json (tcp)
  - enc(key) json (tcp/udp)
  - enc(token) json (udp)
How to gain Independence

Xiaomi Cloud

Copyright: 20th Century Fox
Proxy cloud communication

Robot intern

compass uart_lds uart_mcu

player
0.0.0.0:6665

wifimgr

RoboController

AppProxy

Dustcloud

Miio_client
(local):54322 (tcp)
0.0.0.0:54321 (udp)

Android/
iPhone App

*.fds.api.xiaomi.com (https)

ot.io.mi.com:80 (tcp)
ott.io.mi.com:8053 (udp)

IPC
plain json (tcp)
enc(key) json (tcp/udp)
enc(token) json (udp)

/etc/hosts
130.83.x.x ot.io.mi.com
130.83.x.x ott.io.mi.com

DEFCON 26 – Dennis Giese
What is Dustcloud?

- Proxy or endpoint server for devices
  - Acts as Xiaomi Cloud emulation
  - Reads traffic in plaintext
  - May send commands to the device
    - Change or suppress commands (e.g. Updates)
- Requirements: Device ID, Cloud Key, DNS Redirection
LETS TAKE A LOOK AT THE PRODUCTS
Products

- ~260 different models supported (WiFi + Zigbee + BLE)
- Depending on selected server location
  - Mainland China
  - Taiwan
  - US
  - ...
  - models not always compatible
- My inventory: ~42 different models
  - 99 devices in total

Values estimated, Mi Home 5.3.13, Mainland China Server
Different architectures

- ARM Cortex-A
- ARM Cortex-M
  - Marvell 88MW30X (integrated WiFi)
  - Mediatek MT7687N (integrated WiFi + BLE)
- MIPS
- Xtensa
  - ESP8266, ESP32 (integrated WiFi)

Focus of this talk
Focus of my binary patching talk @IoT Village today

“Why I hate ESP8266” @IoT Village today
Operation Systems

• „Full Linux“ e.g. Ubuntu 14.04
  – Vacuum cleaning robots

• OpenWRT
  – Xiaomi Wifi Speaker, Routers, Minij washing machine

• Embedded Linux
  – IP cameras

• RTOS
  – Lightbulbs, ceiling lights, light strips
## Implementations

<table>
<thead>
<tr>
<th></th>
<th>Vacuum Robot</th>
<th>Smart Home Gateway*</th>
<th>Philips Ceiling Light</th>
<th>Yeelink Bedside Lamp</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Manufacturer</strong></td>
<td>Rockrobo</td>
<td>Lumi United</td>
<td>Yeelight</td>
<td></td>
</tr>
<tr>
<td><strong>MCU</strong></td>
<td>Allwinner + STM + TI</td>
<td>Marvell (Wi-Fi)</td>
<td>MediaTek (Wi-Fi + BLE)</td>
<td></td>
</tr>
<tr>
<td><strong>Firmware Update</strong></td>
<td>Encrypted + HTTPS</td>
<td>Not Encrypted (No SSL stack!)</td>
<td>Not Encrypted + HTTPS (No Cert check!)</td>
<td></td>
</tr>
<tr>
<td><strong>Debug Interfaces</strong></td>
<td>Protected</td>
<td>Available</td>
<td>Available</td>
<td></td>
</tr>
</tbody>
</table>

*Does not apply for DGNWG03LM (Gateway model for Taiwan)

**Bonus:** Chinese device, but unknown communication to server in Salt Lake City, USA (166.70.53.160)
Good news

• Vendors/Developers are lazy
• Assumed development of firmware:
  – Take SDK/toolchain
  – Modify sample that the product runs
  – If it works: publish firmware

  All firmwares very similar (memory layout, functions, strings, etc)
LETS GET ACCESS TO THE DEVICES
Warranty seal?
AQARA SMART IP CAMERA
Overview Hardware

- CPU: Hi3518EV200
  - ARM Cortex-A
- RAM: 64MB
- Flash: 16MByte
- Wi-Fi: Mediatek MT7601UN via USB
- OS: Embedded Linux
- Zigbee-MCU: NXP JN5169
Devices connected via Zigbee

Zigbee (NXP JN5169) based
• Motion Sensor
• Temperature sensors
• Power Plug
• Smoke Detectors
• Smart Door Lock
• ...

DEFCON 26 – Dennis Giese
Serial port after bricking device
Leaked information

- JFFS2 filesystem not properly cleaned
- 3 different credentials from development devices leaked

```bash
0004cc10 e3 b5 3b e8 00 2c 23 20 63 61 74 20 2f 65 74 63 |..;..,# cat /etc|
0004cc20 2f 6d 69 6f 64 65 76 69 63 68 69 73 64 69 6e 67 6e |/miio/device.con|
0004cc30 66 0a 23 20 6d 69 69 6f 2f 64 65 76 69 63 68 69 73 64 |f.# did must be|
0004cc40 23 0a 6d 61 63 3d 32 38 3a 63 6f 6e 73 74 72 69 6e 67 0a |a unsigned int.#|
0004cc50 6b 65 79 3d 4e 41 37 4e 69 6d 4b 6f 50 00 00 4e 73 74 72 69 6e 67 0a |key)p..Nstring.|
0004cc60 66 0a 23 20 6d 6f 64 65 6c 20 6d 61 78 20 6c 65 6e 20 32 33 0a |
x
0004cc70 63 61 6d 65 72 61 2e 61 71 31 0a 70 32 70 5f 69 64 32 0a |camera.aq1.p2p_i|
0004cc80 3a 32 38 3a 64 6c 75 6d 0a 23 20 6d 75 73 74 20 62 65 20 61 74 20 75 6e 73 69 67 6e 65 64 20 69 6e 74 0a |
0004cc90 32 38 63 3a 30 31 31 31 41 0a 11 00 00 03 00 00 03 30 31 31 31 41 0a 11 00 00 |
d=A,...0111A....
```
Rooting

• Serial was not necessary
  – open telnet server (port 23)
  – hardcoded root password in /etc/shadow
    • “root:IlfCcCAiKWPNs:17333:0:99999:7::”
    • DES-Crypt -> password truncated to 8 chars
    • Password: “lumi-201”
  – Same credentials for all cameras
Modifications

• Replace Chinese sound files
• Replace telnetd by dropbear (SSH)
• Change root password
• Replace Camera Software
WI-FI NETWORK SPEAKER
Overview Hardware

- CPU: Amlogic Meson3
  - ARM Cortex-A
- RAM: 128MB
- Flash: 8GByte
- WI-Fi+BT: Broadcom BCM4345
- OS: OpenWRT
  - Samba 3.x
- Released: End 2016
Serial Port

ttyS0 TX

ttyS0 RX
Rooting

• Teardown of device not necessary
• Classic vulnerability: no input validation

http://{ip}:9999/{ssdp id}/Upnp/resource/sys?command=nslookup&host=`echo 192.168.0.2`&dns_server=`/etc/init.d/ssh start`
Firmware updates

• Query Update Information over HTTP

• Firmware updates over HTTP
  – packed LZMA in XML format
  – EXT2 images
  – No signatures
VACUUM CLEANING ROBOTS
Gen 1 Device Overview

Laser Distance Sensor (LDS)
Wall Sensor
Dustbin Sensor
Collision Sensor
Ultrasonic Radar Sensor
Electronic Compass
Cliff Sensor
Gyroscope / Accelerometer
Drop Sensor
Speedmeter
Fan Speed Sensor

Source: Xiaomi advertisement
Teardown
Frontside layout mainboard

512 MB RAM

4GB eMMC Flash

WiFi Module

STM32 MCU

R16 SOC

DEFCON 26 – Dennis Giese
Backside layout mainboard

- R16 UART (115200 baud)
- STM UART (921600 baud)
- LIDAR UART
Frontside layout mainboard (Gen2)

- R16 SOC
- 512 MB RAM
- STM32 MCU
- 4GB eMMC Flash
- WiFi Module
Rooting

• Usual (possibly destructive) way to retrieve the firmware
Rooting

Our weapon of choice:
Rooting (Gen1 + Gen2)

- Shortcut the MMC data lines
- SoC falls back to FEL mode
- Load + Execute tool in RAM
  - Via USB connector
  - Dump MMC flash
  - Modify image
  - Rewrite image to flash
Software

- Ubuntu 14.04.3 LTS (Kernel 3.4.xxx)
  - Mostly untouched, patched on a regular base
- Player 3.10-svn
  - Open-Source Cross-platform robot device interface & server
- Proprietary software (/opt/rockrobo)
  - Custom adbd-version
- iptables firewall enabled (IPv4!)
  - Blocks Port 22 (SSHd) + Port 6665 (player)
  - Fail: IPv6 not blocked at all
Available data on device

- Data
  - Logfiles (syslogs, stats, Wi-Fi credentials)
  - Maps
- Data is uploaded to cloud
- Factory reset
  - Does not delete data: Maps, Logs still exist

~100 Gbyte writes per Year
Available data on device

- Maps
  - Created by player
  - 1024px * 1024px
  - 1px = 5cm
## eMMC Layout

<table>
<thead>
<tr>
<th>Label</th>
<th>Content</th>
<th>Size in MByte</th>
</tr>
</thead>
<tbody>
<tr>
<td>boot-res</td>
<td>bitmaps &amp; some wav files</td>
<td>8</td>
</tr>
<tr>
<td>env</td>
<td>uboot cmd line</td>
<td>16</td>
</tr>
<tr>
<td>app</td>
<td>device.conf (DID, key, MAC), adb.conf, vinda</td>
<td>16</td>
</tr>
<tr>
<td>recovery</td>
<td>fallback copy of OS</td>
<td>512</td>
</tr>
<tr>
<td>system_a</td>
<td>copy of OS (active by default)</td>
<td>512</td>
</tr>
<tr>
<td>system_b</td>
<td>copy of OS (passive by default)</td>
<td>512</td>
</tr>
<tr>
<td>Download</td>
<td>temporary unpacked OS update</td>
<td>528</td>
</tr>
<tr>
<td>reserve</td>
<td>config + calibration files, blackbox.db</td>
<td>16</td>
</tr>
<tr>
<td>UDISK/Data</td>
<td>logs, maps, pcap files</td>
<td>~1900</td>
</tr>
</tbody>
</table>
Update process

1. encrypted packet with pkg info

2. Download [app_url]

Update root pw in /etc/shadow

Decrypt + image OK?

Unpack + dd

Download

Data

system_a

system_b

milO.ota ("mode")

Active copy

rebooting ...

DEFCON 26 – Dennis Giese
Firmware updates

• Integrity
  – MD5 provided by cloud

• Full images
  – Encrypted tar.gz archives
  – Contains disk.img with 512 Mbyte ext4-filesystem

• Encryption
  – Ccrypt [256-bit Rijndael encryption (AES)]
  – Static password: “rockrobo”
<table>
<thead>
<tr>
<th>Address</th>
<th>Length</th>
<th>Type</th>
<th>String</th>
</tr>
</thead>
<tbody>
<tr>
<td>.rodata:0001A...</td>
<td>00000010</td>
<td>C</td>
<td>FormatPartition</td>
</tr>
<tr>
<td>.rodata:0001A...</td>
<td>00000015</td>
<td>C</td>
<td>ChangeShadowPassword</td>
</tr>
<tr>
<td>.rodata:0001A...</td>
<td>0000002C</td>
<td>C</td>
<td>Failed to delete directory '%s'. errno = %d</td>
</tr>
<tr>
<td>.rodata:0001A...</td>
<td>00000027</td>
<td>C</td>
<td>Failed to delete file '%s'. errno = %d</td>
</tr>
<tr>
<td>.rodata:0001A...</td>
<td>00000008</td>
<td>C</td>
<td>CMD&gt; %s</td>
</tr>
<tr>
<td>.rodata:0001A...</td>
<td>00000014</td>
<td>C</td>
<td>%s &gt; /dev/null 2&gt;&amp;1</td>
</tr>
<tr>
<td>.rodata:0001A...</td>
<td>00000017</td>
<td>C</td>
<td>Executing &quot;%s&quot; failed!</td>
</tr>
<tr>
<td>.rodata:0001A...</td>
<td>00000029</td>
<td>C</td>
<td>Computed package MD5 = %s; Expected = %s</td>
</tr>
<tr>
<td>.rodata:0001A...</td>
<td>00000013</td>
<td>C</td>
<td>crypt -d -K %s %s</td>
</tr>
<tr>
<td>.rodata:0001A...</td>
<td>00000009</td>
<td>C</td>
<td>rockrobo</td>
</tr>
<tr>
<td>.rodata:0001A...</td>
<td>00000012</td>
<td>C</td>
<td>Decrypting %s ...</td>
</tr>
<tr>
<td>.rodata:0001A...</td>
<td>00000012</td>
<td>C</td>
<td>Decryption failed</td>
</tr>
<tr>
<td>.rodata:0001A...</td>
<td>0000001F</td>
<td>C</td>
<td>tar xzf %s</td>
</tr>
<tr>
<td>.rodata:0001A...</td>
<td>00000022</td>
<td>C</td>
<td>Extracting image '%s' to '%s' ...</td>
</tr>
<tr>
<td>.rodata:0001A...</td>
<td>0000000F</td>
<td>C</td>
<td>Extract failed</td>
</tr>
<tr>
<td>.rodata:0001A...</td>
<td>00000010</td>
<td>C</td>
<td>tar tf %s &quot;%s&quot;</td>
</tr>
</tbody>
</table>
Let's root remotely

• Preparation: Rebuild Firmware
  – Include authorized_keys
  – Remove iptables rule for sshd
• Send „miIO.ota“ command to vacuum
  – Encrypted with token
    • From app or unprovisioned state
  – Pointing to own http server
Let's root remotely

unprovisioned state

„Get Token“

„milO.ota“

Webserver
SSH

```
login as: root
Authenticating with public key "rsa-key-gami" from agent
Welcome to Ubuntu 14.04.3 LTS (GNU/Linux 3.14.39 armv71)

* Documentation: https://help.ubuntu.com/
Last login: Thu Dec 14 01:43:59 2017 from 192.168.8.67
root@rockrobo:~# 
```
root@rockrobo:~

root@rockrobo:~# apt-get update
Ign http://us.ports.ubuntu.com trusty InRelease
Get:1 http://us.ports.ubuntu.com trusty-updates InRelease [65.9 kB]
Get:2 http://us.ports.ubuntu.com trusty-security InRelease [65.9 kB]
Hit http://us.ports.ubuntu.com trusty Release
Hit http://us.ports.ubuntu.com trusty Release
Hit http://ppa.launchpad.net trusty InRelease
Get:3 http://us.ports.ubuntu.com trusty-updates/main Sources [409 kB]
Get:4 http://us.ports.ubuntu.com trusty-updates/restricted Sources [6322 B]
Get:5 http://us.ports.ubuntu.com trusty-updates/main armhf Packages [875 kB]
Hit http://ppa.launchpad.net trusty/main armhf Packages
Get:6 http://us.ports.ubuntu.com trusty-updates/restricted armhf Packages [8931 B]
Hit http://ppa.launchpad.net trusty/main Translation-en
Get:9 http://us.ports.ubuntu.com trusty-security/main Sources [147 kB]
Get:10 http://us.ports.ubuntu.com trusty-security/restricted Sources [4931 B]
Get:11 http://us.ports.ubuntu.com trusty-security/main armhf Packages [575 kB]
Get:12 http://us.ports.ubuntu.com trusty-security/restricted armhf Packages [8931 B]
Possible Countermeasures

- Changing the firmware key
  - Useless -> we will figure out ;)
- Encrypting/Obfuscating the log-files and maps
  - Recently introduced
  - Here is the AES128CBC-key: “RoCKR0B0@BEIJING”
How to get the log and map AES key?

• RRlogd uses AES encryption functions from OpenSSL library
  – Imported as dynamic library
  – Interesting function: EVP_EncryptInit_ex(...)  

• Helpful tool: ltrace
  – Intercepts library calls
  – Shows contents arguments of function calls
**Persistance**

- Patch the recovery partition
  - Replace custom adbd with open source one
  - disable firewall
- Disable updates
  - Kill SysUpdate process
  - Disable Ccrypt
- Extract credentials
  - Content of “vinda.conf” = root password (XOR 0x37)
  - DID, cloud key
Side note about Entropy

• Recap: Token is AES256 key
  – Method used for Generation:
    • Initialization: srand(seed)
      – seed has $2^{31}$ states
    • 16 times rand()
Summary of the Vacuum

• Rooting
  – **Remote**! (No „foil attack“ required anymore)

• Cloud Connection
  – Run **without** cloud
    • Support by third-party tools (e.g. FloleVac, FHEM, etc)
  – Run with your **own** cloud
HAVING FUN IN HACKING
Connection to the Dark Side

• Idea by Prof. Noubir: Let’s run Tor hidden services on IoT
  – Paper from 2015: OnionBots, a stealthy botnet with compromised IoT devices
• Easy to install in Ubuntu
  – Make SSH accessible via TOR
  – No need for NAT ;)

OnionBots: Subverting Privacy Infrastructure for Cyber Attacks

Amirali Sanatini
Northeastern University
amirali@ccs.neu.edu

Guevara Noubir
Northeastern University
noubir@ccs.neu.edu

Abstract—Over the last decade, botnets have evolved by adopting a sequence of increasingly sophisticated strategies to evade detection and take over, and to monetize their infrastructure. At the same time, the success of botnets’ infrastructure is at risk level of this arms-race. We contend that the next wave of botnets will rely on subverting privacy infrastructure and a non-trivial use of cryptographic mechanisms. The
Using empty space
Using empty space

• Zigbee module fits in vacuum
  – Use serial connection
  – ARM binaries of Gateway run natively
  – Result: Zombie-Gateway-Vacuum

• USB stick
  – More space: mobile Data storage
  – Soldered to MicroUSB port
Mobile Wi-Fi mapper

• Idea:
  – Parsing of position2d from player logfile
    \{x\_pos, y\_pos, yaw\_pos, x\_vel, y\_vel, yaw\_vel\}
  – Retrieving WiFi information from Linux kernel
    \{link, level, noice, SSID, BSSID\}

• Developed with Andrew Tu @HackBeanpot 2018, Boston
Mobile Wi-Fi mapper

Genuine + Jack Morton Office, NE Side, 5th floor
Mobile Wi-Fi mapper
If in need of additional space

- Done by Dustcloud user
- Reason: broken MMC-Chip
- Not recommended for everyone ;)

```
mmc: MMCClock 50000000, MMCbuswidth 4
[mmc]: SD/MMC Card: 4bit, capacity: 7600MB
[mmc]: boot0 capacity: 0kB, boot1 capacity: 0kB
[mmc]:/MMC2 init: OK!!!
```
If in need of additional space
IoT chatting with IoT
One word of warning...

• Never leave your devices unprovisioned
  – Someone else can provision it for you
    • Install malicious firmware
• Be careful with used devices
  – e.g. Amazon Marketplace, Ebay, etc.
  – Some malicious software may be installed
• Never install rooted firmware from untrusted sources !!!!
  – Especially not from Russian forums!
Conclusion

• Basic best practices not used
  – firmware signatures 😞
  – HTTPS, certificate verification 😞
  – Hardware security features 😞
• Good
  – We can modify the devices
• Bad
  – Someone else can do too
Acknowledgements

• Daniel Wegemer (aka DanielAW)
• Prof. Guevara Noubir (CCIS, Northeastern University)

Northeastern University
College of Computer and Information Science

• Secure Mobile Networking (SEEMOO) Labs and CROSSING S1

Secure Mobile Networking
CROSSING

• Andrew Sellars and Team (Boston University Technology & Cyberlaw Clinic)

School of Law
Technology & Cyberlaw Clinic

http://www.ccs.neu.edu/home/noubir/
https://www.seemoo.informatik.tu-darmstadt.de/
https://sites.bu.edu/tclc/
Questions?
Meet me at the IoT Village here at Defcon

Contact:
See: http://dontvacuum.me
Telegram: https://t.me/kuchenmonster
Twitter: dgi_DE
Meet me in Boston/@DC617