

AN EXPERIMENTAL STUDY OF SECURITY AND PRIVACY OF THE INTERNET OF THINGS DEVICES

BY

Strahinja Trećakov

Bachelor of Science in Computer Science

New Mexico State University

Las Cruces New Mexico

December 2016

ACKNOWLEDGMENTS

I would like to thank my advisor, Satyajayant Misra, for his support, and patience. Also I would like to thank Sunka Sachin, for working with me on some parts of this research. Special thank to Travis Mick who was always there to hear issues and give me a hint on this research. Finally, I would like to thank my family for their continuous support in my endeavors.

ABSTRACT

As the Internet of Things (IoT) market size is rapidly increasing everyday, Cisco predicted that there would be 50 to 200 billion connected devices by 2020[19]. Living in such world privacy and security should be a major concern. IoT devices even today fail to encrypt at least some of the traffic that they send and receive. Many of the devices exchange personal or private information with servers on the Internet in the clear text, completely unencrypted. In this research, I studied the behavior of a few IoT devices, specifically the August Smart Lock, the Chromecast, the TrackR Bravo, the Wink Light bulbs and a hub, and the Monster wireless speaker. More specifically, I monitor the network and study the packets that go to the network. I present the security and privacy weaknesses of each device and a possible solution to it. I conclude this paper with the overview of the research and scope of future work.

CONTENTS

LIST OF TABLES.....	v
LIST OF FIGURES.....	vi
INTRODUCTION.....	1
RELATED WORK.....	2
PROBLEM DEFINITION.....	3
UBERTOOTH ONE.....	4
CRACKLE.....	6
WIRESHARK.....	7
ANDROID BLUETOOTH SNIFFING.....	7
SYSTEM MODEL AND ASSUMPTIONS	8
DESIGN AND EVALUATION.....	9
CHROMECAST.....	9
AUGUST SMART LOCK	13
WINK LIGHT HUB AND BULBS.....	14
TRACKR BRAVO	15
MONSTER ROCKIN' ROLLER SPEAKER.....	17
CONCLUSION	18
REFERENCES.....	18

LIST OF TABLES

Table 1: Table of Commands	5
Table 2: Table of Specifications.....	8

LIST OF FIGURES

Figure 1: The Ubertooth One.....	4
Figure 2: Chromecast website leak.....	10
Figure 3: Chromecast detailed leak	10
Figure 4: Chromecast command leak	11
Figure 5: Chromecast smartphone leak.....	11
Figure 6: Chromecast smartphone app leak.....	12
Figure 7: August Smart Lock QUIC.....	14
Figure 8: TrackR Bravo LTK leak.....	16
Figure 9: Monster Rockin' Roller speaker	17

INTRODUCTION

The home is becoming a place where all devices are going to be “smart”, and connected to the wireless home network. I will be manipulating these devices remotely, through smartphones. IoT devices use sensors (cameras, microphones, thermal detectors, etc.) and actuators (lights, speakers, locks), that brings the security and privacy concerns. Most of these devices use encryption, however, many of them have some security or privacy omissions that can be used against the customers. For example, some of the IoT devices give out the location of the user [13], others can give up the exact location of a user in the house [2]. Then there are security issues that can lead hackers to take over the control of the device. However, attackers can also hack into the home network and on that way they can control some IoT devices. In this paper, I study the behavior of a few IoT devices, specifically the August Smart lock, the Chromecast, the Wallet TrackR, the Wink light bulb and hub, and the Monster wireless speaker. Based on the evaluation, I show the lack of privacy and security of these devices. I used these few tools Wireshark, Ubertooth One, Crackle, and Android 4.4+, to sniff and analyze packets exchanged between devices.

RELATED WORK

Today, IoT is a field that is developing really fast. Every day some company introduces a new IoT device to make our lives and homes, easier and safer. However, new technologies bring some concerns. The biggest security issues of IoT devices are physical attack, Trojan attacks, virus damage, keys decryption, DOS, and traffic analysis [6]. This shows us that every single IoT device brings us a security risk [4]. Researchers are trying to find a solution to these security and privacy issues. Some of them tested a couple of IoT devices and found out that some devices are sending decrypted packets to the network, which in some cases was the information of the user [1], [2], [14]. In [16] they were able to get your pins and password, just by sniffing the connection between a smart watch and a smartphone. One of the biggest discoveries, is that some of the wireless keyboards do not encrypt the packets at all[17]. I use Wireshark packet analyzer to capture the network activity and analyze the vulnerabilities [14]. Wireshark can peer inside the network and examine the details of traffic at a many of levels, ranging from connection level information to the bits comprising a single packet.to monitor and analyze the network in a home full of IoT devices and investigate the information that is being leaked by these devices and conclude how secure are they [11].

Most of the related papers propose some high-level architecture solution or only address the problem. I am going to point out the weaknesses of these IoT devices and try to find a unique solution for all of them and on that way propose our solution for the new IoT devices.

PROBLEM DEFINITION

The goal of IoT devices is to communicate the data from the devices, over the network remotely and to monitor them. If the communication between such devices is not secure enough, then a big risk is posed for the consumer. Basically, these devices are vulnerable in many forms such as authentication, identity, access control, network security, encryption, malicious data transfer etc. I have selected some devices that are getting more commonly used today.

August Smart lock, Chromecast, TrackR Bravo, Wink light bulb and hub, and Monster wireless speaker. For all these devices installed in our lab, I have capture and the monitor the network activity using the Wireshark packet analyzer.

UBERTOOTH ONE

The Ubertooth One is a sniffer on a hardware platform with an RF frontend, CC2400 radio chip, and LPC microcontroller[18]. The CC2400 radio chip can monitor a single BTLE channel because it has a reconfigurable radio transceiver. It can also partially sniff the classic Bluetooth, but only using LCP microcontroller. The Ubertooth one has 2.4 GHz transmit power and receive sensitivity to a Class 1 Bluetooth device. It has a standard Cortex debug connector and in-system programming serial connector. On the Ubertooth board, there are also expansion connector and six LED indicators.

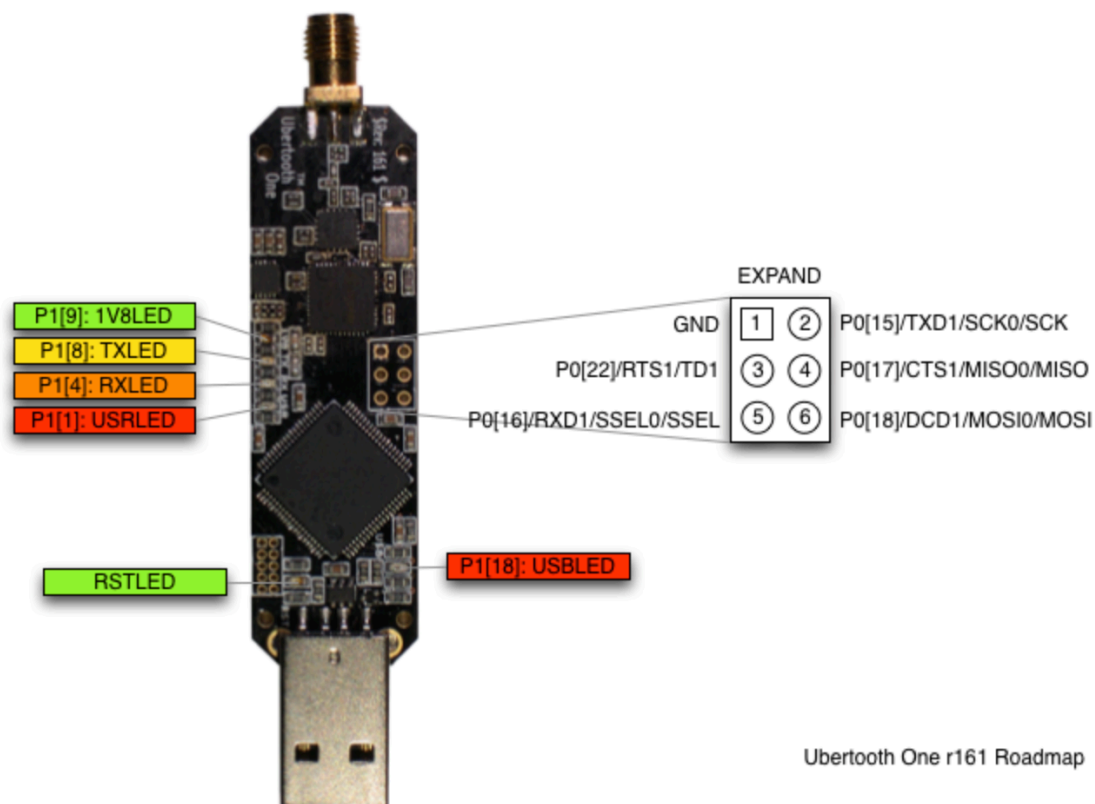


Figure 1: The Ubertooth One

LED guide:

- . RST: indicates that the LPC175x is powered on.
- . 1V8: indicates that the CC2400 is being supplied with 1.8 V. Control of this supply depends on firmware. 1V8 power is required to activate the crystal oscillator that is required to activate USB.
- . USB: indicates that USB has passed enumeration and configuration.
- . TX: Control of this LED depends on firmware. It typically indicates radio transmission.
- . RX: Control of this LED depends on firmware. It typically indicates radio reception.
- . USR: Control of this LED depends on firmware.

The TX, RX, and USR LEDs blink in a distinctive chasing pattern when the bootloader is ready to accept USB DFU commands.

Table 1: Table of Commands

Command	Description	Power draw (amps)
	Idle	0.09 A
<code>ubertooth-dump</code>	Receive	0.13 A
<code>ubertooth-util -t</code>	Transmit	0.22 A
<code>ubertooth-dfu --write</code>	Firmware upgrade	0.10 A

Since I was capturing BTLE packets, I used `ubertooth-btle -f -c filename.pcap`. This command will save the sniffed packets into .pcap file format.

CRACKLE

Crackle is a open source software written by Mike Ryan to decrypt BTLE encryption. Since BTLE uses AES-CCM[20], the only way to encrypt the packet was to get the Long Term Key(LTK) before master and a slave establish an encrypted session[23]. The key exchange protocol starts by picking a Temporary Key(TK), a 128 bit AES key. The TK depends on pairing mode. Since there are three defined pairing modes, Just Works(0), 6-digit pin(128 bit value between 0 and 999999) and OOB(128 bit value exchanged out of band). If devices use Just works or 6 digit PIN, we can use brute force algorithm to guess TK. Since we have TK, it is easy to get Short term key(STK) since TK is used to encrypt STK. Finally STK is used to establish link-layer encryption over which Long Term Key(LTK) is exchanged. From there master and a slave are communicating using LTK encryption, which is useless[25]. It is only working with PPI and BLUETOOTH_HCI_H4_WITH_PHDR frames.

WIRESHARK

Wireshark is a free, world's widely used network protocol analyzer[21]. It is similar to tcpdump, with some sorting and filtering plugins. It is very powerful piece of software to do network analysis, because of its filters and decryption options[22].

ANDROID BLUETOOTH SNIFFING

Android introduced Bluetooth sniffing capability on versions 4.4(Kit Kat) and higher. To capture the exchanged files between your phone and some other Bluetooth device, you have to go to Setting, Developer options, and the mark Enable Bluetooth HCI snoop log. This will save captured files in to btsnoop_hci.log file on your phone. Then you can open that using Wireshark on your computer.

SYSTEM MODEL AND ASSUMPTIONS

Table 2: Table of Specifications

DEVICE	Wink Light bulbs	August Smart Lock	TrackR Bravo	Monster wireless speaker	Chromecast
POWER	12 WATTS	4AA batteries	CR1616 battery	Power	Power
LIFETIME	25000 hours	1 year	1 year	N/A	N/A
CONNECTIVITY	Wi-Fi 2.4 and 5GHz	Wi-Fi 2.4 and 5GHz and Bluetooth	Bluetooth Low Energy(BTLE)	Bluetooth v3.0	Wi-Fi 2.4 and 5GHz
CONTROL	Dimmable, ON, OFF	LOCK, UNLOCK	ON,OFF	Volume control	Play, Pause, Rewind, forward
APP	Wink	August	TrackR	N/A	TVCast(C), YouTube...

DESIGN AND EVALUATION

CHROMECAST

The Google Chromecast is a media streaming device that goes into HDMI port on your TV. It can be connected over your home wireless network to smartphone, table, and laptops. Your device will work as a remote and you can stream TV shows, movies, music, sports, games and more.

Testing

I preformed tests on Chromecast on 2 different setups. First the Chromecast and a smartphone were connected to the same wireless home network. Then used MacBook Pro as a controller for Cromecast on the same home network.

After capturing and analyzing the data, I found out that it actually shows some vulnerabilities.

The first setup, leaked and discovered information's:

The IP address of the Chromecast device, it's name, uuid, bssid, country, MAC address, Chromecast pin code. It also gives us the public key, time zone, ssid of the home network, Operating System and it's version installed on the remote device. Furthermore, I was able to find from which website is the streaming coming from and the stop and play commands.

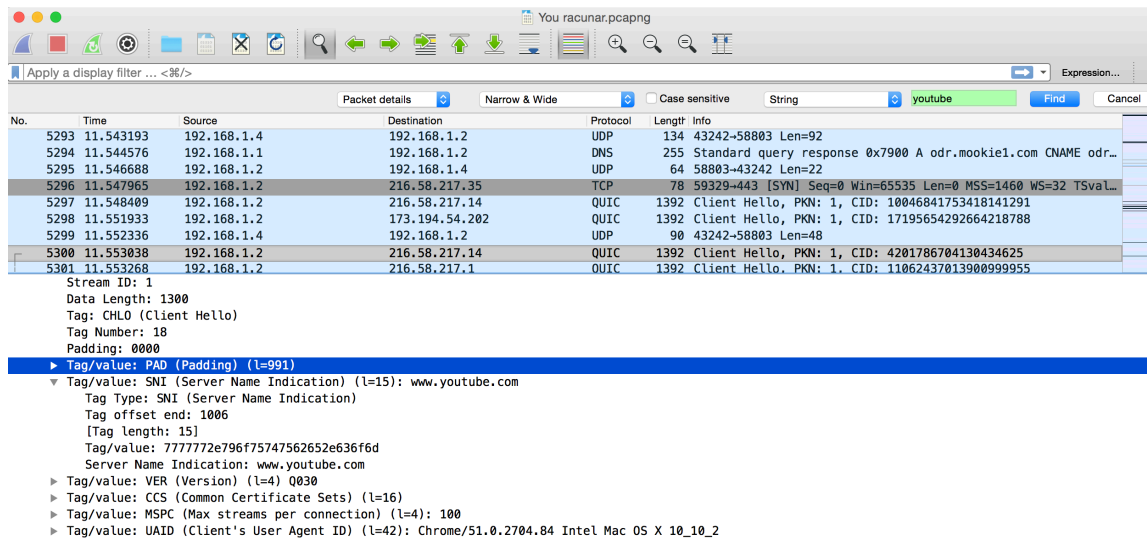


Figure 2: Chromecast website leak

```
GET /setup/eureka_info HTTP/1.1
Host: 192.168.1.4:8008
Connection: keep-alive
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_2) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/51.0.2704.106 Safari/537.36
Accept: */*
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8

HTTP/1.1 200 OK
Access-Control-Allow-Headers:Content-Type
Cache-Control:no-cache
Content-Length:1376
Content-Type:application/json

{"bssid":"84:1b:5e:2a:05:a8","build_version":"63621","cast_build_revision":"1.19a.
63621","closed_caption":
{"connected":true,"ethernet_connected":false,"has_update":false,"hotspot_bssid":"FA:8F:CA:5D:
1B:F6","ip_address":"192.168.1.4","locale":"en-US","location":{"country_code":"US","latitude":
255.0,"longitude":255.0},"mac_address":"A4:77:33:DB:
44:80","name":"StraleCast","noise_level":-88,"opencast_pin_code":"1440","opt_in":
{"crash":true,"location":false,"opencast":true,"stats":true},"public_key":"MIIBCgKCAQEArse61+d8qXQ
CVERfKwTm5bG0iCiRapkb6g8MGfsxSp+y+QH1vBtA3XsxzkAYnUS9CoHB40dn6INX9q/KA41N/
nnp8PIvCpj98H0JAbDD6PTlm1LmQaqtUcEdoETqvdIwr0gBQGM93Xb5JI/Ey3hj8lWIpQuZ1FC/
5UVEDfpKBka709+ia4iMAJAvy5Cz7cQk8FChk/+yeece57yXoeL5HHnYmfhBKZ+6Cpt3g/pDvDhqWJdL/+0kFM/irRp+
+sWbRy018yJDKdH/hMhGDy02M/Nqs3wXItIQ/4xc+pzmFGwrDo+F0Ex1Wo5P/
exgdLymBzuGveXQaxkZc4C2pgXwIDAQAB","release_track":"stable-channel","setup_state":60,"setup_stats":
{"historically_succeeded":true,"num_check_connectivity":0,"num_connect_wifi":
0,"num_connected_wifi_not_saved":0,"num_initial_eureka_info":0,"num_obtain_ip":
0},"signal_level":-43,"ssdp_udn":"75flae90-7bed-ae97-cc8f-722b7658a065","ssid":"Strale-
Network","time_format":1,"timezone":"America/
Denver","tos_accepted":true,"uma_client_id":"FBF715CF-1C47-40F7-8896-BB49760E0B87","uptime":
279.31,"version":7,"wpa_configured":true,"wpa_id":1,"wpa_state":10}
```

Figure 3: Chromecast detailed leak


```

GET /apps/YouTube HTTP/1.1
Host: 192.168.1.4:8008
Connection: keep-alive
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_10_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/51.0.2704.84 Safari/537.36
Accept: */*
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-US,en;q=0.8

HTTP/1.1 200 OK
Content-Length:228
Content-Type:application/xml

<?xml version="1.0" encoding="UTF-8"?>
<service xmlns="urn:dial-multiscreen-org:schemas:dial">
  <name>YouTube</name>
  <options allowStop="true"/>
  <state>stopped</state>
  <discovery>DIAL,CastV2</discovery>
</service>

```

Figure 4: Chromecast command leak

The second setup, leaked following information's:

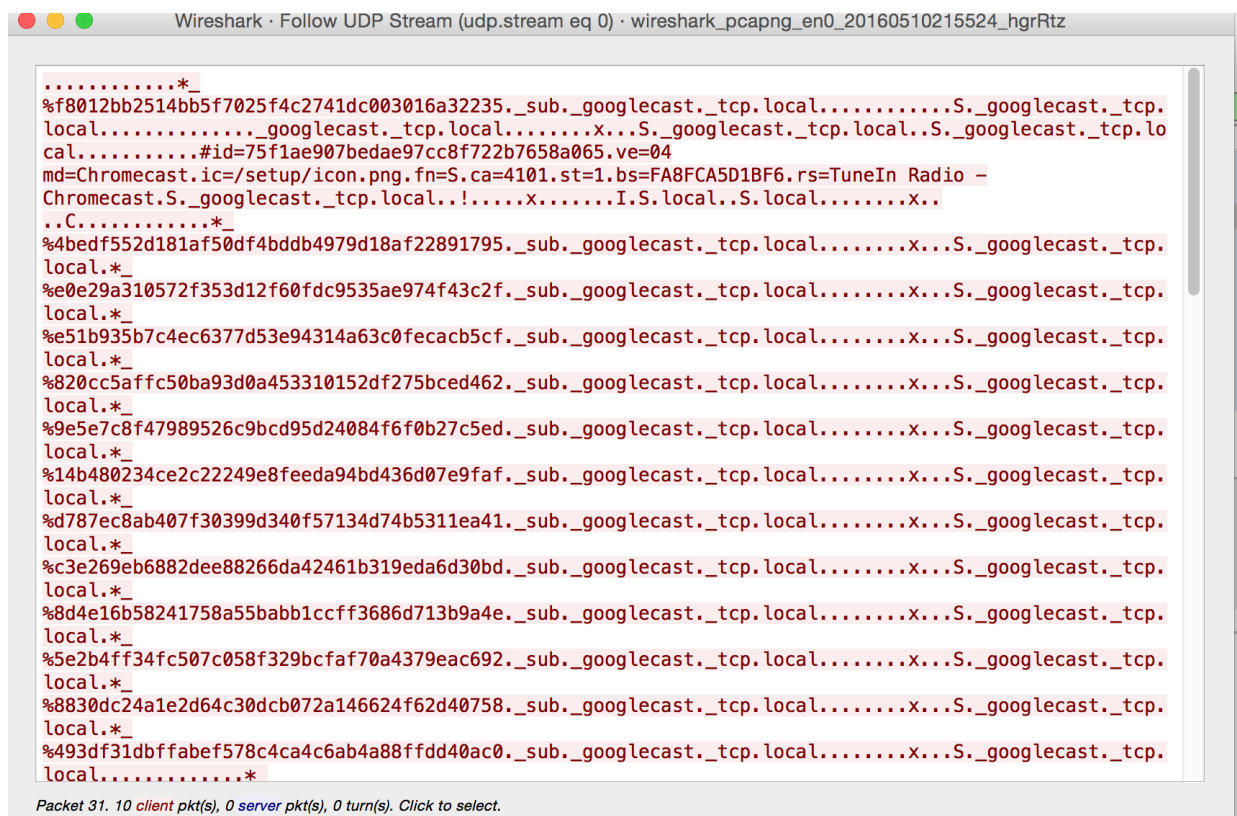
The IP address of the Chromecast device, it's name, uuid, and MAC address. Packets that were captured also show what app did I use on my phone to stream it.

```

fn=StraleCast.ca=4101.st=1.bs=FA8FCA5D1BF6
rs=YouTube TV+Chromecast-75f1ae907bedae97cc8f722b7658a065._googlecast._tcp.local..!.....x.2....I
$75f1ae90-7bed-ae97-cc8f-722b7658a065.local.$75f1ae90-7bed-ae97-cc8f-722b7658a065.local.....x..
..C....._googlecast._tcp.local.....x.D
+Chromecast-75f1ae907bedae97cc8f722b7658a065._googlecast._tcp.local.
+Chromecast-75f1ae907bedae97cc8f722b7658a065._googlecast._tcp.local.....#id=75f1ae907bedae97c
c8f722b7658a065.rm=28F06C38A07DD64E.ve=05
md=Chromecast.ic=/setup/icon.png

```

Figure 5: Chromecast smartphone leak



```
.....*  
%f8012bb2514bb5f7025f4c2741dc003016a32235._sub._googlecast._tcp.local.....S._googlecast._tcp.  
local....._googlecast._tcp.local.....X...S._googlecast._tcp.local..S._googlecast._tcp.lo  
cal.....#id=75f1ae907bedae97cc8f722b7658a065.ve=04  
md=Chromecast.ic=/setup/icon.png.fn=S.ca=4101.st=1.bs=FA8FCA5D1BF6.rs=TuneIn Radio -  
Chromecast.S._googlecast._tcp.local..!.....X.....I.S.local..S.local.....X..  
..C.....*  
%4bedf552d181af50df4bddb4979d18af22891795._sub._googlecast._tcp.local.....X...S._googlecast._tcp.  
local.*_  
%e0e29a310572f353d12f60fdc9535ae974f43c2f._sub._googlecast._tcp.local.....X...S._googlecast._tcp.  
local.*_  
%e51b935b7c4ec6377d53e94314a63c0fecacb5cf._sub._googlecast._tcp.local.....X...S._googlecast._tcp.  
local.*_  
%820cc5affc50ba93d0a453310152df275bcd462._sub._googlecast._tcp.local.....X...S._googlecast._tcp.  
local.*_  
%9e5e7c8f47989526c9bcd95d24084f6f0b27c5ed._sub._googlecast._tcp.local.....X...S._googlecast._tcp.  
local.*_  
%14b480234ce2c2249e8feeda94bd436d07e9faf._sub._googlecast._tcp.local.....X...S._googlecast._tcp.  
local.*_  
%d787ec8ab407f30399d340f57134d74b5311ea41._sub._googlecast._tcp.local.....X...S._googlecast._tcp.  
local.*_  
%c3e269eb6882dee88266da42461b319eda6d30bd._sub._googlecast._tcp.local.....X...S._googlecast._tcp.  
local.*_  
%8d4e16b58241758a55babb1ccff3686d713b9a4e._sub._googlecast._tcp.local.....X...S._googlecast._tcp.  
local.*_  
%5e2b4ff34fc507c058f329bcfaf70a4379eac692._sub._googlecast._tcp.local.....X...S._googlecast._tcp.  
local.*_  
%8830dc24a1e2d64c30dcb072a146624f62d40758._sub._googlecast._tcp.local.....X...S._googlecast._tcp.  
local.*_  
%493df31dbffabef578c4ca4c6ab4a88ffdd40ac0._sub._googlecast._tcp.local.....X...S._googlecast._tcp.  
local.....*  
  
Packet 31. 10 client pkt(s), 0 server pkt(s), 0 turn(s). Click to select.
```

Figure 6: Chromecast smartphone app leak

All these informations were sent through clear-text over HTTP. In both settings Chromecast and a remote device communicate over QUIC protocol. This raises some privacy concerns. Both devices send a SSDP message to IP address 239.255.255.250 on port 1900. First device sends a M-SEARCH HTTP/1.1 packets for Chromecast discovery, while Chromecast sends HTTP/ 1.1 OK message to the device.

AUGUST SMART LOCK

August Smart Lock is an electromechanical lock that performs unlocking and locking operations on a door when it receives these instructions from a smartphone. It can work thru Bluetooth connection or wireless connection using a hub or both. In all cases, you need a smartphone with an August app, where you give commands to unlock or lock. You can also give an access to your friend for a period of time that you want. You will also get a notification when someone unlocks or locks the door. Another feature is to set an automatic unlocking or locking when you are close by your door. The August Smart Lock can be connected to other IoT devices and on that way make your home smarter.

Testing

For the August Smart Lock I used next few setups. First, I monitored the Bluetooth connection between the August Smart Lock and an Android smartphone. I captured the packets using the Android's default setting. After analyzing packets I did not find any privacy leaks. Then I connected the smart lock to the hub, which was connected to the home network. The phone was first connected to the same home network as the hub, and after that, I monitored the network when the phone was connected to the different network. To capture all data that was going to the home network I used Wireshark. After studying the data, I did not find any privacy leaks.

From a screenshot of the packets analyzed with Wireshark, we can see that August Smart Lock uses QUIC protocol for communication, and that data is encrypted.

and a smartphone. I was unable to capture any data from hub to light bulbs since it uses ZigBee wave.

TRACKR BRAVO

The TrackR Bravo is a small device that can help you locate or track your lost item up to 100feet distance. It is a coin size device, with replaceable CR1620 battery that lasts up to 1 year. It uses Bluetooth Low Energy to connect to Android or iOS device. In case you lose your item that have trackR Bravo attached and it is out of range from your smart phone, if any other trackR Bravo user's smart phone come within 100 feet of your item, the trackR server will notify you with GPS location of where your item was last seen.

Testing

The TrackR uses only Bluetooth connection that I capture thru Android's default option of capture the Bluetooth packet stream and using the Ubertooth One. After capturing packets that were exchanged between the smart phone device and TrackR Bravo, I used Wireshark to analyze them. The .pcap file captured by Android phone shows some vulnerabilities of this device. I was able to find names of devices, their addresses, encryption details(Long Term Key(LTK), Signature Key, Identity Resolving Key, Stored Link Key(SLK), the alter request(commands). This data is unencrypted and exchanged in a plain text between devices prior to being encrypted by Bluetooth chip.

The second test I preformed was with Ubertooth One. I sniffed packed exchanged between trackR Bravo and Android smartphone. After storing all the packets into a .pcap file, I was not able to use Crackle to decrypt the communication. I also contacted person who wrote it and it

seems that Ubertooth one is not always capable of capturing all the packets. He suggested me to recapture couple of times and try to decryption using Crackle, but id did not work.

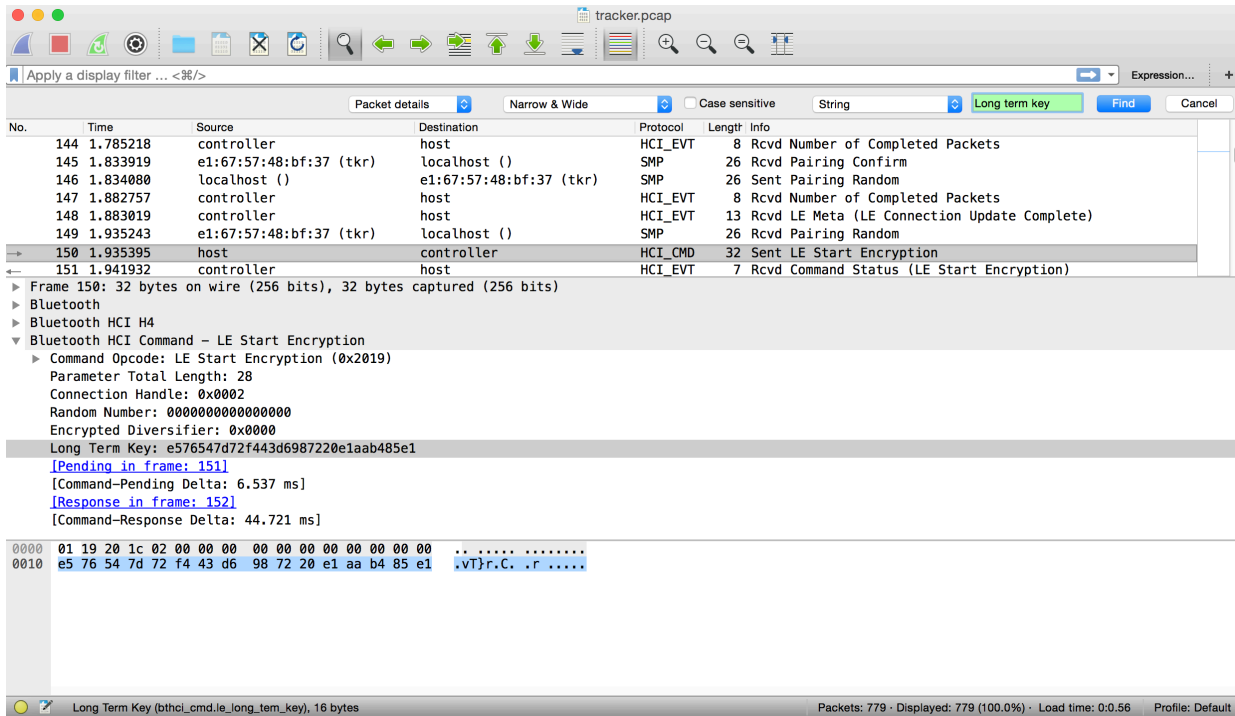


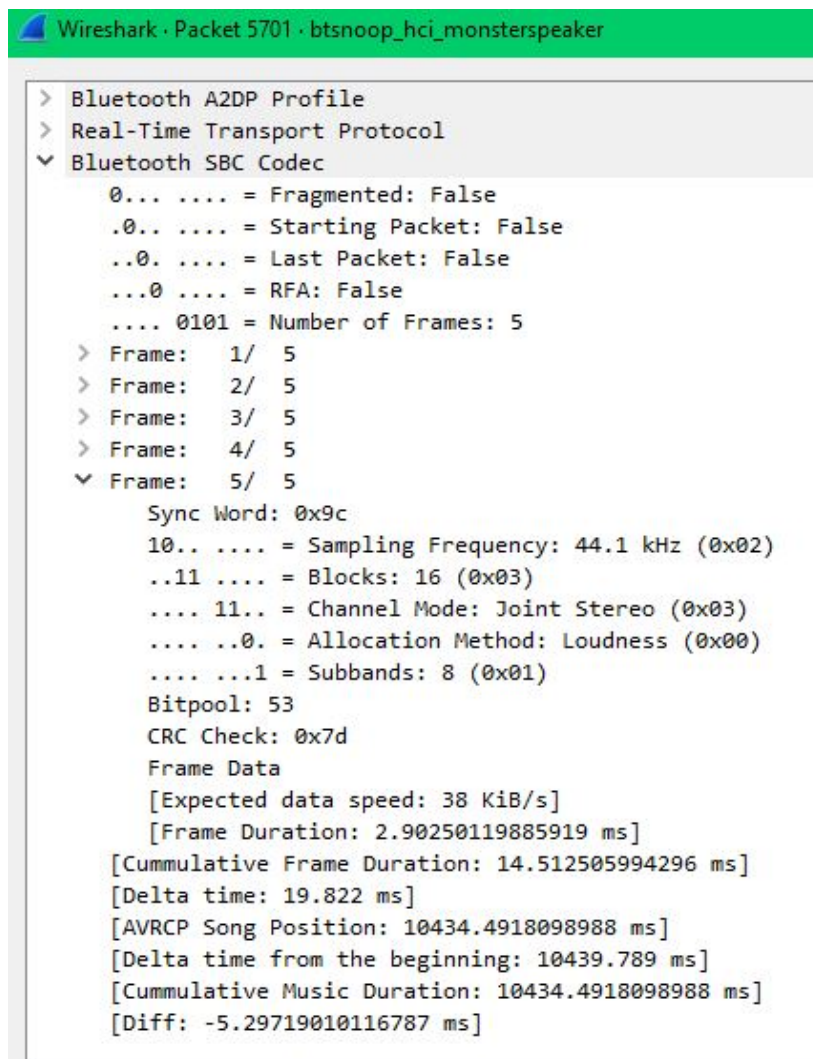
Figure 8: TrackR Bravo LTK leak

MONSTER ROCKIN' ROLLER SPEAKER

The Monster Rockin' Roller wireless speaker has connectivity option thru Bluetooth or Aux cable.

Testing

I captured the Bluetooth packets via Android phone and after analyzing, I find out that it shows the name of the speaker and that the sender was transmitting a music file.



The image shows a Wireshark packet capture window with the title bar 'Wireshark · Packet 5701 · btsnoop_hci_monsterspeaker'. The packet list on the left shows the following hierarchy: Bluetooth A2DP Profile, Real-Time Transport Protocol, and Bluetooth SBC Codec (expanded). The packet details pane on the right shows the following fields:

- 0... = Fragmented: False
- .0.. = Starting Packet: False
- ..0. = Last Packet: False
- ...0 = RFA: False
- 0101 = Number of Frames: 5
- > Frame: 1/ 5
- > Frame: 2/ 5
- > Frame: 3/ 5
- > Frame: 4/ 5
- ▼ Frame: 5/ 5
 - Sync Word: 0x9c
 - 10.. = Sampling Frequency: 44.1 kHz (0x02)
 - ..11 = Blocks: 16 (0x03)
 - 11.. = Channel Mode: Joint Stereo (0x03)
 -0. = Allocation Method: Loudness (0x00)
 -1 = Subbands: 8 (0x01)
 - Bitpool: 53
 - CRC Check: 0x7d
 - Frame Data
 - [Expected data speed: 38 KiB/s]
 - [Frame Duration: 2.90250119885919 ms]
 - [Cumulative Frame Duration: 14.512505994296 ms]
 - [Delta time: 19.822 ms]
 - [AVRCP Song Position: 10434.4918098988 ms]
 - [Delta time from the beginning: 10439.789 ms]
 - [Cumulative Music Duration: 10434.4918098988 ms]
 - [Diff: -5.29719010116787 ms]

Figure 9: Monster Rockin' Roller speaker

CONCLUSION

My study motivates the need to critically analyze the privacy concerns that expose the details of a consumer who is in a IoT network. First, I have discussed the privacy threats that prevail in the current IoT market. Secondly, I picked the IoT devices(the August Smart Lock, the Chromecast, the TrackR Bravo, the Wink Light bulbs and a hub, and the Monster wireless speaker) and the tools(Wireshark, Ubertooth One, Crackle and Android 4.4+) that I need to test for the leakage of information from them. Then, I have built an architecture for running the tests on these devices. Next, I have examined the tests and found out that some devices are very secure, the information leaked by them is highly encrypted whereas devices such as chrome cast and monster speakers leak sensitive information of the consumers such as the device names, the websites being cast, shared keys, and many more. Finally, some technical challenges are discussed on how to capture the network traffic of IoT communication that provide directions for future research.

REFERENCES

- [1] N. Feamster, "Who Will Secure the Internet of Things?" <https://freedom-to-tinker.com/blog/feamster/who-will-secure-the-internet-of-things/>
- [2] P. N. Tracy, "Security flaw compromises location of Nest Thermostat owners" <http://www.dailydot.com/technology/nest-security-flaw/>
- [3] Ning, Huansheng, and Hong Liu. "Cyber-physical-social based security architecture for future internet of things." *Advances in Internet of Things 2*, no. 01 (2012): 1. <http://file.scirp.org/Html/17031.html>
- [4] Hwang, Yong Ho. "IoT security & privacy: threats and challenges." In *Proceedings of the 1st ACM Workshop on IoT Privacy, Trust, and Security*, pp. 1-1. ACM, 2015. <http://dl.acm.org/citation.cfm?id=2732216>
- [5] Riahi, Arbia, Yacine Challal, Enrico Natalizio, Zied Chtourou, and Abdelmadjid Bouabdallah. "A systemic approach for IoT security." In *Distributed Computing in Sensor Systems (DCOSS), 2013 IEEE International Conference on*, pp. 351-355. IEEE, 2013. http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6569455&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxppls%2Fabs_all.jsp%3Farnumber%3D6569455
- [6] Gang, Gan, Lu Zeyong, and Jiang Jun. "Internet of things security analysis." In *Internet Technology and Applications (iTAP), 2011 International Conference on*, pp. 1-4. IEEE, 2011. <http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6006307&url=http%3A%2F%2Fieeexplore.ieee.org%2Fstamp%2Fstamp.jsp%3Farnumber%3D6006307>
- [7] Roman, Rodrigo, Jianying Zhou, and Javier Lopez. "On the features and challenges of security and privacy in distributed internet of things." *Computer Networks* 57, no. 10 (2013): 2266-2279. https://www.ftc.gov/sites/default/files/documents/public_comments/2013/07/00031-86244.pdf
- [8] Babar, Sachin, Parikshit Mahalle, Antonietta Stango, Neeli Prasad, and Ramjee Prasad. "Proposed security model and threat taxonomy for the internet of things (IoT)." In *Recent Trends in Network Security and Applications*, pp. 420-429. Springer Berlin Heidelberg, 2010.
- [9] The Internet of Things: Security Research Study <https://www.veracode.com/sites/default/files/Resources/Whitepapers/internet-of-things-whitepaper.pdf>
- [10] Princeton presentation on IoT security http://www.cs.princeton.edu/~sgrover/sgrover_files/papers/privacycon2016-grover.pdf

- [11] Wireshark tutorial on IoT sniffing
<http://www.freaklabs.org/index.php/tutorials/software/sniffing-the-internet-of-things-with-wireshark-sensniff-and-freaklabs.html>
- [12] Conti, Mauro, Luigi Vincenzo Mancini, Riccardo Spolaor, and Nino Vincenzo Verde. "Analyzing Android Encrypted Network Traffic to Identify User Actions." *Information Forensics and Security, IEEE Transactions on* 11, no. 1 (2016): 114-125.
<http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7265055>
- [13] Who is tracking your run and bike activity tracking app privacy issues investigated
<https://www.pentestpartners.com/blog/who-is-tracking-your-run-run-and-bike-activity-tracking-app-privacy-issues-investigated/>
- [14] Notra, Sukhvir, Muhammad Siddiqi, Hassan Habibi Gharakheili, Vijay Sivaraman, and Roksana Boreli. "An experimental study of security and privacy risks with emerging household appliances." In *Communications and Network Security (CNS), 2014 IEEE Conference on*, pp. 79-84. IEEE, 2014.
- [15] Tips to Test Network Performance of Wearables
<http://developerboards.att.lithium.com/t5/AT-T-Developer-Program-Blogs/Tips-to-Test-Network-Performance-of-Wearables/ba-p/38661>
- [16] Wang, Chen, Xiaonan Guo, Yan Wang, Yingying Chen, and Bo Liu. "Friend or Foe?: Your Wearable Devices Reveal Your Personal PIN." In *Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security*, pp. 189-200. ACM, 2016.
- [17] Wireless keyboard sniffing
<https://threatpost.com/keysniiffer-vulnerability-opens-wireless-keyboards-to-snooping/119461/>
- [18] <https://github.com/greatscottgadgets/ubertooh/wiki/Ubertooh-One>
- [19] <http://www.fool.com/investing/general/2016/01/18/internet-of-things-in-2016-6-stats-everyone-should.aspx>
- [20] <https://github.com/mikeryan/crackle>
- [21] <https://wiki.wireshark.org>
- [22] <https://www.wireshark.org>
- [23] http://www.ellisys.com/technology/een_bt09.pdf

- [24] <http://gizmodo.com/august-smart-lock-and-connect-review-ill-use-keys-tha-1685319060>
- [25] Ryan, Mike. "Bluetooth: with low energy comes low security." In *Presented as part of the 7th USENIX Workshop on Offensive Technologies*. 2013.
- [26] <https://hackerific.net/2012/01/28/Spectrum-Tools-and-Ubertooth-One/>
- [27] <http://cerescontrols.com/tutorials-3/sniffing-bluetooth-packets-with-kismet-and-wireshark-in-ubuntu-12-04/>
- [28] <https://www.security-sleuth.com/sleuth-blog/2015/9/6/now-i-wanna-sniff-some-bluetooth-sniffing-and-cracking-bluetooth-with-the-ubertoothone>
- [29] <http://www.tcpdump.org/linktypes.html>