A Survey of Various Methods for Analyzing the Amazon Echo

Ike Clinton
School of Mathematics and Computer Science
The Citadel, The Military College of South Carolina
Charleston, South Carolina 29409
Email: iclinton@citadel.edu

Lance Cook
School of Mathematics and Computer Science
The Citadel, The Military College of South Carolina
Charleston, South Carolina 29409
Email: lcook2@citadel.edu

Dr. Shankar Banik
School of Mathematics and Computer Science
The Citadel, The Military College of South Carolina
Charleston, South Carolina 29409
Email: shankar.banik@citadel.edu

I. Introduction

Internet of things, a buzzword in todays media, is defined as a proposed development of the Internet in which everyday objects have network connectivity, allowing them to send and receive data. If we look past the buzzwords though what it really is, is a vast and rapidly growing frontier of new technology that includes a variety of smart devices. It is the network of cyber-physical systems or things embedded with electronics, software, sensors, and connectivity. These embedded systems can refer to a wide range of devices ranging from smart TVs and network connected light switches to pace makers and portable health monitoring equipment such as the Fitbit. Our research sets out to look at how this system of interconnected smart devices affects our privacy, what implications it might have on the global Internet community, and how can we show this with a practical analysis and vulnerability analysis of our own smart devices. In our research we test the Amazon Echo.

A. What are Smart Devices

The first smart devices started appearing around the 1970s. For example, the first ATM ubiquitous money dispensers were first online in 1974. Today, using our smart phones we can lock and unlock doors remotely. The range of what these smart devices are is very wide, but what remains a constant is that the Internet of things is an exploding market made up of over a billion smart devices today. During 2008, the number of things connected to the Internet exceed the number of people on earth, and it is estimated by the year 2020 there will be over 50 billion smart devices. This is just the beginning though. The next step for all of these connected smart devices is for them to begin to start talking to each other. Ciscos blog gave us a great example scenario of this. Imagine that you have a meeting that is then pushed back 45 minutes which is then communicated to your car, but your car knows it will need gas to make it to the train station which will take 5 minutes for you to fill up. It is also communicated to your clock that there was an accident on your route to the train station causing a 15 minute detour. Not to mention that your train is running 20 minutes behind schedule. All these devices talking to each other add this up, and your alarm clock allows you to sleep in an extra 5 minutes. It also signals your coffee maker to turn on 5 minutes late as well so you are ready to go for the day. The scenario given to us by Cisco is in fact well within our reach very soon. In fact by the end of 2011, 20 typical households generated more Internet traffic then the entire Internet did combined in 2008.

II. The Amazon Echo

The Amazon Echo [1] is a smart hub device that allows you to use voice commands to control it and other smart devices connected to it. The Echo connects to Alexa, a cloud-based voice service, to allow you to use voice commands to command the device to do a multitude of things from answering basic questions like weather and news to controlling the lights in your house. The Echo's Alexa is very similar to Apple's Siri and Androids' Cortana. Our research set out to analyze the privacy model of the Echo, and to become familiar with the intended use cases of the device. Before buying the device we started by trying to study past and current research, and that was part of the driving reason why we chose to explore the Echo. We were unable to find very much if any research into the security model of the device. For our research we wanted to analyze the default configurations of the device, obtain firmware through various methods, analyze the firmware and determine potential vulnerabilities, and finally test the exploitation techniques we found as potential vulnerabilities on the device. Instead of first breaking our personal Echo apart, as it is an expensive device, we found a tear down and analysis online. Using the tear down we were able to find a complete breakdown of the Amazon Echo.

A. Research and Information Gathering

Our project began with a basic fact-finding hunt for information. We wanted to find out anything and everything we could about the device. We scoured the Internet for resources from official documentation, to developer forums, to user message boards, as well as others. We set out to learn everything that we could about the device so that we knew its ins and outs and inner workings.

B. Device Setup and Use

Our first step in this process was to unbox the Echo, set it up just like a normal user would, and monitor the communications the device made during the setup process. We found that the Echo uses a simple web server for device setup. This is typical of many IoT devices. The user connects to the WiFi signal that the Echo emits, sets the device up using the phone app, and connects the Echo to the user's own home WiFi. The device then acquires an IP address via DHCP, and can then be used as normal assuming the WiFi provides Internet connectivity to the Echo.

C. Software Used

In our research we used a multitude of programs some of which we were familiar with, and others that we had to learn specifically for this project. The first more familiar program that we used was Nmap. Nmap allowed for us to do network scanning of the device when it showed up on our network. This program is how we discovered which ports where open on the devices. We also used Wireshark to analyze the traffic between the Echo and Amazon's servers. Little research effort was put into this area, however as most of the focus was on hardware based methods. Putty was used as a serial terminal for reading the boot logs from the Echo. Finally, various other Linux development tools and packages were used in addition to the TI Developer CPU SDK.

On the Amazon echo we had to learn a whole new methodology for analyzing vulnerabilities. This is discussed later in the paper, but we learned how to use a soldering iron and a USB to serial adapter to read the data coming out of the UART port on the bottom of the Echo. This proved critical in understanding the boot process of the device, among other things.

D. Hardware Used

In our research, we used a variety of odds and ends found in our lab to aid us in our research. In particular, we used a soldering iron, male to male connecting wires, a USB-TTL serial UART cable (we found the JBtek Windows 8 Supported Debug Cable drivers worked best), as well as some stripped USB cable wiring. We found the small, braided wire was best for doing the miniscule and detailed soldering work that needed to be done.

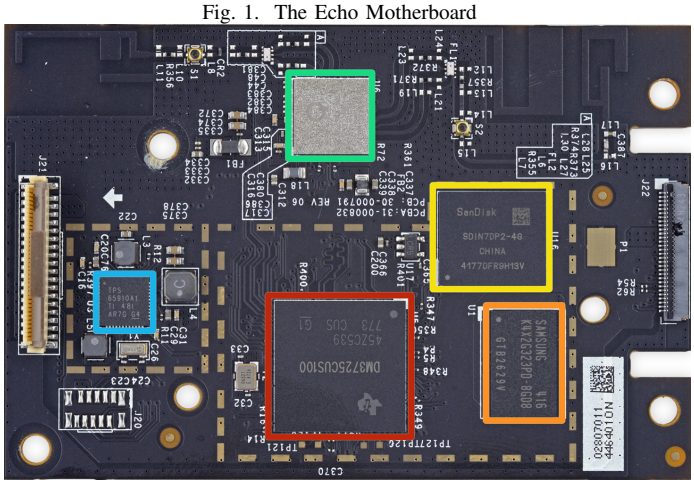
III. Echo Hardware - iFixit

iFixit [2] is the self-proclaimed "Free Repair Manual" and was a great resource in our project. On this site, our team found instructions for the disassembly and teardown of the device. It also provided valuable information on how the different components of the device worked, and were a great starting point for reverse engineering the hardware.

Figure 1 shows the Amazon Echo motherboard.

- The Red square is an Texas Instruments DM3725 Digital Media Processor.
- The yellow is the Echo's 4GB SanDisk SDIN7DP2 4GB iNAND flash memory chip.
- The orange is a Samsung K4X2G323PD-8GD8 256 MB LPDDR1 RAM chip.
- The green chip is a Qualcomm Atheros QCA6234X-AM2D Wi-Fi and Bluetooth Module.
- Finally the blue chip is a Texas Instruments TPS65910A1 Integrated Power Management IC.

Fig. 1. The Echo Motherboard



Other points of interests are the ribbon cable connectors at J21 and J22 (pictured far left and far right in Figure 1). These connecting sites are used to connect the speaker/microphone board (top board inside the Echo), and the power and speaker driver board (on the bottom of the device). The bottom power and speaker driver board was the main focus of our research and reverse engineering.

IV. Hardware Reverse Engineering

We began our research into exploiting the echo hardware by looking at common techniques that have been used to exploit other devices. Specifically, we wanted to know if there had been other Amazon specific devices that had hardware root methods available. The thinking was that the devices may have similar design flaws, vulnerabilities, or attack surfaces. One such discovered method is detailed below.

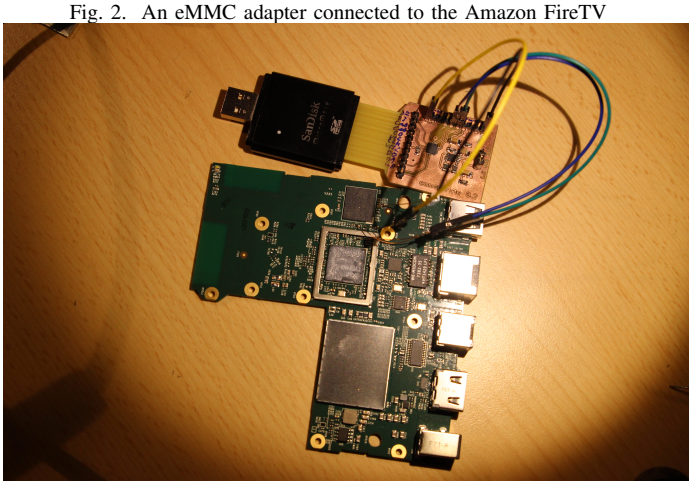
A. eMMC Root

One possible approach that we identified would be to use an eMMC style root like what has been done with the Amazon FireTV in the past. We did not have the time to test this avenue as the pinout for this on the board would have to be identified if it is even present. This still could be a promising avenue for any interested researchers in the future. A quick google search brings up much relevant information and tutorials about this method. An example of an owner performing this technique is shown in figure 2.

B. JTAG

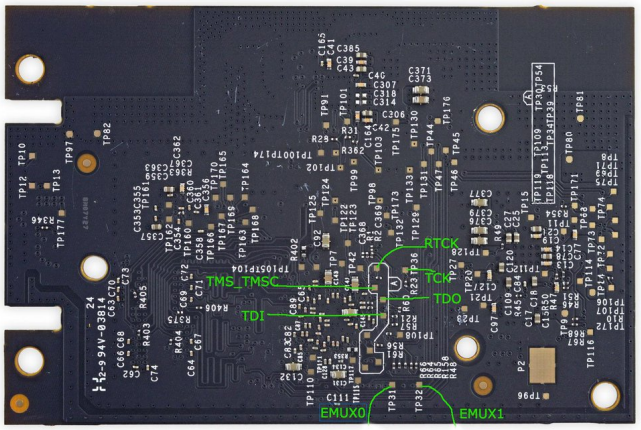
A twitter user by the name of Bill Finlayson handle "bill_billbill_" posted to twitter what he thinks to be the JTAG (Joint Test Action Group) pinout of the device. This could allow access to debugging and programming features on the

Fig. 2. An eMMC adapter connected to the Amazon FireTV



Echo that normally aren't available. Our team did not have the ability to purchase JTAG debugging equipment for our research but this is a promising avenue that could allow full control of the device if successful.

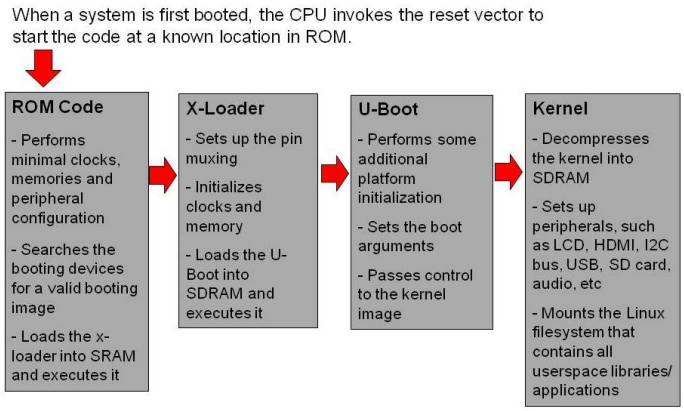
Fig. 3. Potential Echo JTAG pinout



C. The Echo Boot Process

Understanding the boot process of the Echo was critical in understanding the hardware, as well as understanding how software (including the Linux Kernel) was loaded into the device. The figure below from OMAPpedia [12] describes the boot process that all similar TI processors use. First, code flashed to the Echo ROM performs some device setup, and searches for a boot device in a programmatically defined way. The echo, in particular, is set to boot from an external SD card device first, and fails-over to the internal eMMC second if no attached SD card is detected. Once a boot device has been selected, it searches the first valid FAT32 partition for a file called "MLO." This is the TI X-Loader program that initializes clocks and memory, and mainly loads U-Boot into SDRAM and executes it. The job of U-Boot is to set boot arguments and then locate the linux kernel, and give the kernel image control of the processor. Once the kernel is validated and executed, peripherals, storage devices, etc. are all initialized and loaded. Any startup scripts such as the Alexa/Echo initialization scripts are also run at this point.

Fig. 4. The echo boot sequence



D. UART Pinout

We decided if we were able to solder in to the ports we could connect it to a UART Serial to USB device. Soldering into the bottom of the Echo is extremely difficult as the pads are very small(about 1x1mm) and requires a steady hand. After a lot of practice we were successful in connecting to the Echo and began reading the output of the device. The output of the device showed us the U-Boot process. It is shown in the pages that follow.

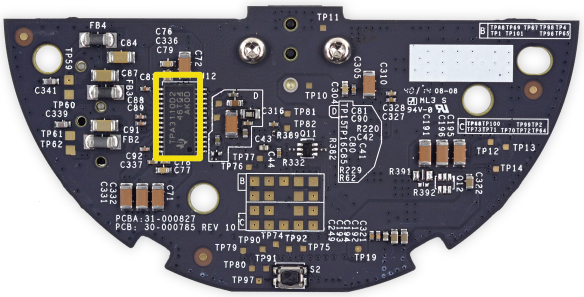
```
    Texas Instruments X-Loader 1.51 (Mar  6 2015 - 04:24:27)
LAB126 Rev 0
Starting X-loader on mmc-0...failed!
Starting X-loader on mmc-0...failed!
Booting from eMMC . . .
Starting X-loader on mmc-1...Reading boot sector
155568 Bytes Read from MMC
Starting OS Bootloader from MMC...
Starting OS Bootloader...(time = 628 ms)

U-Boot 2010.06-00001-g65e5723 (Jun 22 2015 - 22:05:52)
OMAP34xx/35xx-GP ES2.1, CPU-OPP2 L3-165MHz
OMAP3 LAB126 board + LPDDR/NAND
I2C:   ready
DRAM:  256 MiB
MMC:   OMAP SD/MMC: 0, OMAP SD/MMC: 1
Using default environment

In:    serial
Out:   serial
Err:   serial
OMAP3 Lab126 Rev: 0x1a
Die ID #{removed by me}
76 bytes read in 6 ms (11.7 KiB/s)
463 bytes read in 5 ms (89.8 KiB/s)
824 bytes read in 7 ms (114.3 KiB/s)
Animation Version = 3
File System is consistent
file found deleting
update journal finished
File System is consistent
update journal finished
Card did not respond to voltage select!
Invalid uuid. Booting by block dev
booting ...main-A
*
Booting from mmc ...
2605272 bytes read in 491 ms (5.1 MiB/s)
## Booting kernel from Legacy Image at 82000000 ...
   Image Name:   Linux-2.6.37
   Image Type:   ARM Linux Kernel Image (uncompressed)
   Data Size:    2605208 Bytes = 2.5 MiB
   Load Address: 80008000
   Entry Point:  80008000
   Verifying Checksum ... OK
   Loading Kernel Image ... OK
OK
Starting kernel ...
[    0.000000] Trying to install type control for IRQ385
[    0.000000] Trying to set irq flags for IRQ385
[    0.158660] mtdoops: mtd device (mtddev=name/number) must be supplied
[    0.168914] ks8851 spi1.0: failed to read device ID
[    0.205841] codec: aic32xx_i2c_probe : snd_soc_register_codec success
[    0.250244] Power Management for TI OMAP3.
[    0.260070] drivers/rtc/hctosys.c: unable to open rtc device (rtc0)
[    2.315979] DSPLINK Module (1.65.01.05_eng) created on Date: Jun 23 2015 Time: 05:09:20
```

```
Shared memory /QSpeakerIn.shm deletion failed.
Shared memory /QEarconIn.shm deletion failed.
Shared memory /AudiodCmd.shm deletion failed.
Shared memory /BMicsOut.shm deletion failed.
Shared memory /BPhoneMic.shm deletion failed.
Shared memory /BTraitReport.shm deletion failed.
Shared memory /BAsrMetadata.shm deletion failed.
Shared memory /BRemoteMic.shm deletion failed.
CGRE[824]: Started the CGroup Rules Engine Daemon.
shared memory /QSpeakerIn.shm created successfully. (byte_num=95232.)
shared memory /QEarconIn.shm created successfully. (byte_num=16000.)
shared memory /AudiodCmd.shm created successfully. (byte_num=3000.)
shared memory /BMicsOut.shm created successfully. (msg_size=2, msg_num=1048575.)
shared memory /BPhoneMic.shm created successfully. (msg_size=2, msg_num=16000.)
shared memory /BRemoteMic.shm created successfully. (msg_size=2, msg_num=16000.)
shared memory /BTraitReport.shm created successfully. (msg_size=24, msg_num=128.)
shared memory /BAsrMetadata.shm created successfully. (msg_size=1, msg_num=131072.)
CMEM Shared Sizes: Audio A2D 9612 82836 Aux A2D 240276 1600276
```

E. Developer/Debug Ports

The main focus of our research was on the debug ports on the bottom of the device. The breakdown of the Echo is what gave us our second area of interest. On the bottom of the echo are located 18 different pins that each serve various purposes. These pins provide 15V power, 3V power (for and SD card), UART TX, RX, and ground, as well as all of the pins necessary for connecting an external SD card.

Fig. 5. The bottom board on the Echo. Debug pads bottom center.



F. MMC Pinout

Our team was able to identify the pinout on the bottom of the device. It is detailed in figure 5. This pinout allows the connection of an external SD card. The echo is configured by default to boot from this device if it detects it during startup. This would allow a custom OS to be loaded onto a card, and run with the Amazon Echo.

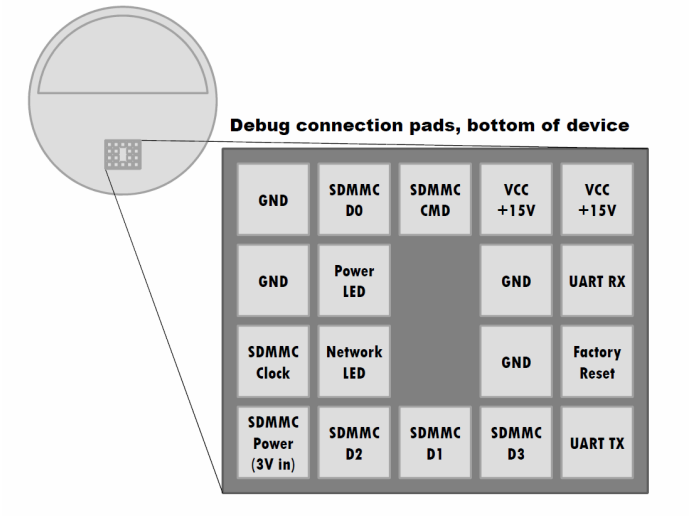
G. SD Card Image Build Process

The final step in taking advantage of the SD card pinout was to configure and SD card that the Echo would boot from. We created a free account on TI's support website [16]and downloaded the DM37x SDK for linux and installed it on an Ubuntu 10.04 LTS (TI indicates this as the recommended version). From there, we used their mk3PartSDCard.sh script (from the TI wiki) [17] to format the card. Once done, we moved the MLO, u-boot.bin, and uImage files over to the boot partition, and copied the linux filesystem to the ext3 partition labeled "rootfs." Once done, our SD card was ready to be booted from by the Echo.

H. 3D Printing an Interface Device

One such approach to interfacing with the SD card pins would be to build a physical device that could hold the

Fig. 6. Pinout for the debug ports



necessary boards and give slots to provide easy connectivity. We got the idea from Amazon's own device pictured below [6] [7]:

Fig. 7. Amazon's Debug Device attached to bottom of Echo

As you can see it appears that Amazon's board was able to nest in the UART port and connect it with Ethernet, and SD Card slots. We were then able to mock up our own possible interfacing device in AutoCAD based of this finding. Our designs are shown below starting with a side view of the base.

Here the side view emphasizes the slots we designed to allow the input of our external peripherals.

Next we have a smaller rectangle piece that would align with the UART port on the bottom of the echo. We would be able to set our pins in this so that we would not need to solder onto the bottom of the Echo. In not doing so it makes it much easier to connect and disconnect the Echo as needed.

Finally, we have a top down view of the model. Here we can see how the device would sit in the mold, and where the

Fig. 8. Side view of a potential device design

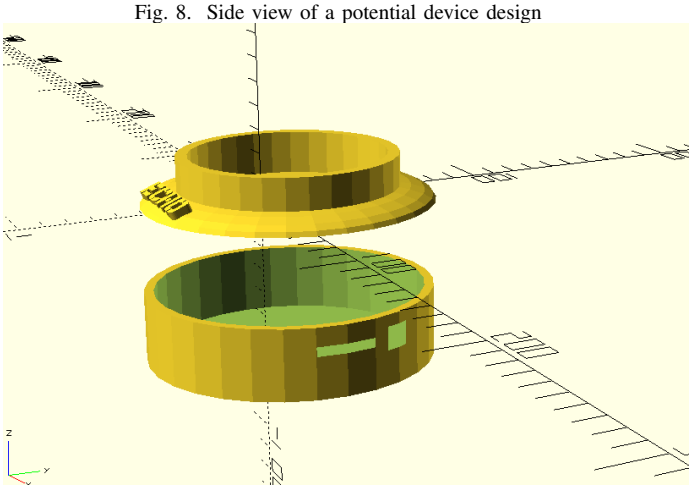


Fig. 9. Rectangle piece used to line up pins

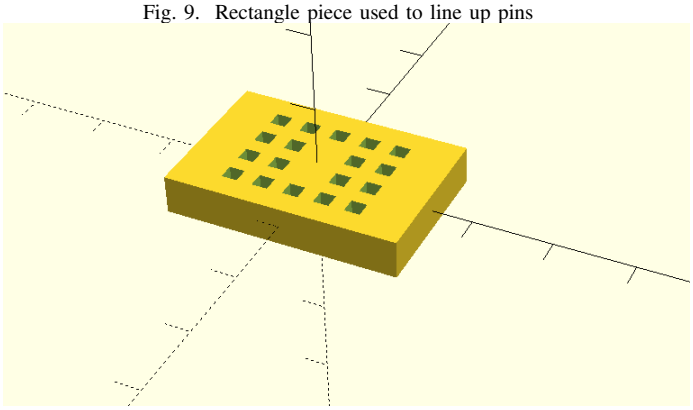
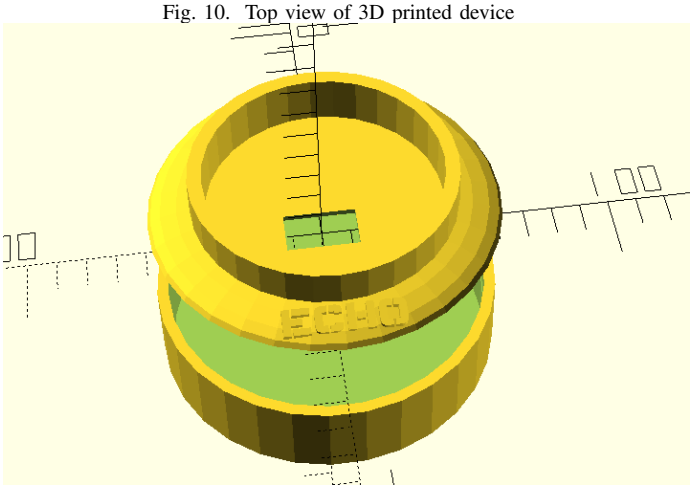


Fig. 10. Top view of 3D printed device



smaller rectangle piece would sit as well. In conclusion, our mock up would allow a much easier connection to the Amazon Echo based off their own interface device. However, due to time constraints we were unable to complete this process.

V. Summary of Findings

Our research focused on identifying and investigating as many different possible attack surfaces as we could. In summary, we identified three main avenues of approach for accessing the device. These are, in order of perceived difficulty (easiest first): the SD card pinout, an eMMC style root, and finally JTAG. We believe that any of these approaches would allow further access into the file system of the Echo that would allow security researchers the ability to reverse engineer binaries for vulnerabilities, scan the device for hardcoded credentials, and much more. This together would allow for a more informed consumer, and a more knowledgeable user to be aware of the privacy and security implications of the Echo. On the other hand, it could also prove that the Echo is indeed quite a robust and secure device. Either way, further analysis is required to confirm or deny any such claims.

VI. Conclusion

All of these things go to show us a few things about the current landscape of embedded devices or more commonly known in the industry as the Internet of Things. For one thing, we did notice that some developers are doing a decent job of keeping track of disclosed vulnerabilities and patching them in subsequent firmware updates. Many, if not all, of their devices still ship factory default with exploitable firmware. Developers should take more care to make sure they arent shipping out exploitable products, and should review basic security concepts like GPG symmetric encryption. The work that we are doing right now in the security of these devices is important for the future of the world we live in. If companies continue to release un-secure products that we will continue to see these findings coming out. The even worse fear is that someone with more malicious intents will find a vulnerability on key devices used by everyone, or the government. A great example of this being the iCloud hack that released a lot of private user information onto the web. The future of the Internet, and our highly technology connected lives depend on these devices to be secure if we would like to see a future like the one Cisco outlined for us. If not we may see the evolution of embedded devices being used to create botnet armys to DDoS website, mine bitcoins, and a multitude of other malicious functions. In fact the first recorded large scale IoT hack has already taken place. Proofpoint found that the compromised gadgets ranging from smart televisions to smart refrigerators were used to send over 750,000 malicious emails to targets between December 26, 2013 and January 6, 2014. A great quote that sums up the importance of the security of these embedded devices, and how the security of them is so crucial to the growing field of interconnectivity is that, If one thing can prevent the Internet of things from transforming the way we live and work, it will be a breakdown in security. In conclusion, we found that the Amazon Echo we found was vulnerable to physical attacks including possible command injections using the UART port, and vulnerable to DDoS during setup. However, general security of the device still remains relatively strong, a big difference compared to other devices that have been researched by us and others in the industry in the past. We hope that our research, and the research of others continues

to bring to light these huge vulnerabilities bringing to light the impending consequences if changes are not made to security of such devices.

REFERENCES

[1] Amazon.com, Inc, *Amazon Echo Product Page* http://www.amazon.com/Amazon-SK705DI-Echo/dp/B00X4WHP5E

[2] iFixit, *iFixit, The Free Repair Manual* https://www.ifixit.com/Teardown/Amazon+Echo+Teardown/33953

[3] Andrew Tang, *A guide to Pentration Testing*, Issue 8. Pages 8-11, ISSN 1353-4858, http://dx.doi.org/10.1016/S1353-4858(14)70079-0, August 2014.

[4] Nuno Antunes and Marco Vieira, *Penetration Testing for Web Services*, Computer 47, no 2. 30-36. Applied Science and Technology Full Text (H.W. Wilson), EBSCOhost (accessed February 23, 2015), February 2014.

[5] Jim Cicconi, *The Internet of Things*, http://blogs.cisco.com/. CISCO, n.d. Web.

[6] CNET, *Alexa, unlock my door:Vivint now works with Amazon Echo: CES 2016* http://www.cnet.com/videos/alexa-unlock-my-door-vivint-now-works-with-amazon-echo/

[7] CNET, *Alexa learned a bunch of new tricks at CES 2016* http://www.cnet.com/pictures/alexa-learned-a-bunch-of-new-tricks-at-ces-2016-pictures/2/

[8] M.J Covington and R. Carskadden, *Threat Implications of the Internet of Things*, pp. 1,12. Cyber Conflict (CyCon), 2013 5th International Conference on, vol., no., pp.1,12, 4-7 June 2013

[9] *Definition of the Internet of Things in English*, Issue 8. Internet of Things: Definition of Internet of Things in Oxford Dictionary (American English) (US). Oxford Dictionaries, n.d. Web. 04 May 2015.

[10] S. Jajodia, P. Ammann, and C.D McCollum, *Survicing Information Warfare Attacks*. Computer, vol.32, no.4, pp.57,63, Apr 1999.

[11] Selena Larson, *The Malware Is Coming from INSIDE THE HOUSE!*, ReadWrite. 16 Jan. 2014. Web. 04 May 2015.

[12] OMAPpedia, *Bootloader Project* http://omappedia.org/wiki/Bootloader_Project

[13] M U Farooq, Muhammad Waseem, Anjum Khairi and Sadia Mazhar, *A Critical Analysis on the Security Concerns of Internet of Things*, International Journal of Computer Applications 111. 1-6, February 2015.

[14] S.M. Sajjad, and M. Yousaf, *Security Analysis of IEEE 802.15.4 MA in the context of Internet of Things*. Information Assurance and Cyber Security (CIACS), 2014 Conference on , vol., no., pp.9,14, 12-13 June 2014.

[15] S.W. Smith, *Cryptographic scalability challenges in the smart grid (extended abstract)*, pp. 1,3 Innovative Smart Grid Technologies (ISGT), 2012 IEEE PES, 16-20 Jan. 2012

[16] Texas Instruments *DM3725 DaVinci Video Processor* http://www.ti.com/product/DM3725

[17] Texas Instruments *Texas Instruments Wiki: How to Make 3 Partition SD Card* http://processors.wiki.ti.com/index.php/How_to_Make_3_Partition_SD_Card

[18] John Yeo, *Feature: Using Penetration Testing to Enhance your Company's Security*, Computer Fraud and Security 2013: 17-20. ScienceDirect, EBSCOhost, April 2013.