Internet of Things Device Penetration Testing

Aimee True
Department of Computer Science
Colorado School of Mines
Golden, CO, USA
true@mymail.mines.edu

Andrew Harelson
Department of Computer Science
Colorado School of Mines
Golden, CO, USA
andrewharelson@mymail.mines.edu

Bradley Helliwell
Department of Computer Science
Colorado School of Mines
Golden, CO, USA
bhelliwell@mymail.mines.edu

Thien Ngo Le
Department of Computer Science
Colorado School of Mines
Golden, CO, USA
thienngole@mymail.mines.edu

***Abstract*— Internet of Things, or IoT, is a trending phrase in the world today. It refers to a system of internet-connected devices that have the ability to exchange data. These devices provide us with many advantages, such as having smart homes and smart cities. However, the price of having the conveniences offered by IoT devices is a potential loss of security and privacy, as IoT devices suffer from a wide range of security vulnerabilities. In this paper, we describe an experimental project performed that analyzed and reported found security vulnerabilities and potential privacy issues in smart home systems. This experiment was performed using a red versus blue team exercise, where we, the red team, attempted to find and exploit vulnerabilities while the blue team attempted to counteract us.**

Keywords—IoT; smart home devices; red team; vulnerabilities; security and privacy;

I. Introduction

Over the past several years, internet of things, or IoT, devices have become increasingly popular. These internet-connected devices have the ability to automatically collect and exchange data with other devices and with users. These devices are also characteristically easy to use and have a vast amount of potential value. IoT devices can be used in larger capacities, such as for large-scale transportation networks, or for more every day, individual use. For individual users, IoT devices can provide a wide variety of conveniences, often in the form of smart home devices. Some examples of services that smart home devices provide are scheduled keeping, health monitoring, and home security. These IoT devices come in all shapes and sizes, from smart speakers to smart vacuums to smart grills.

However, despite the wide array of opportunities that come with IoT devices, the devices also come with security and privacy concerns. Malicious attacks that leverage the use of IoT devices have already begun to occur. For example, the Mirai Botnet DDoS attack occurred in 2016 and affected large parts of the United States and Europe. With all these devices connected to each other and the internet, there are concerns for protecting user's privacy and information. Part of the appeal of IoT devices is the fact that they are constantly collecting and storing information collected from users and their homes. Intimate and private data can be collected and if that information is not securely stored, attackers can easily capture it. Privacy has always been a great concern in cybersecurity, but IoT devices are especially concerning. For example, smart cameras such as the Ring video doorbell records videos outside of a user's house. A dedicated attacker could gain access to those videos and could use them to deduce the address of their victim and use it to exploit them in a variety of different ways.

Despite these concerns, security measures for IoT devices are not being implemented. One of the big reasons for the lack of security is a desire to innovate quickly and minimize costs. Due to the rising popularity of these devices, new types and versions of devices are getting released almost every day. Therefore, to gain competitive advantages, developers try and get products out to the public as quickly as possible. As a result, developers do not spend the time needed to ensure that their IoT devices have effective security measures. Developers are not required to implement these security measures due to a lack of enforced security standards. In addition, implementing security measures can be difficult due to the fact that IoT devices have simple hardware.

The primary goal of this project was to learn about vulnerabilities in IoT devices and demonstrate how they can be implemented. This is done in an effort to help protect IoT devices and prove that those security concerns are valid. We researched different vulnerabilities and used them to perform attacks on smart home devices in a realistic setting. Because we are all novices in this field, we used vulnerabilities already discovered by others. The real value in our work is demonstrating that these vulnerabilities still exist and that non-experts can perform these attacks. By finding vulnerabilities that can be easily exploited by a beginner team, the devices can be better secured and protected in the future.

II. Approach

A. Key Idea

To find flaws and vulnerabilities in IoT devices, and determine how to fix those vulnerabilities, this experiment was carried out in a blue team versus red team style fashion. The red team—our team—focused on attacking the devices, whereas the blue team focused on defense.

B. Experimental Setup

To perform this experiment, we set up a lab environment with a dedicated Linksys router. The OpenWrt operating system was installed on the router, which allowed for extensive customization of the router. The setup also contained an Android phone and an Ubuntu desktop computer. The computer contained a number of tools installed upon it, including

VirtualBox through which a virtual machine of Kali Linux was installed. Most importantly, a number of smart home devices were set up in the lab. This list contained the following: an indoor DropCam, Belkin WeMo smart switch, Amazon Echo, Samsung SmartThings, Foscam R4, Lefun Camera, Withings sleep monitor, Omron 7 series, Ring, August Smart Lock Pro, Wemo smart plug, VOCOlinc PM2 Smart Power Strip, Netmato weather station, Triby speaker, Nest Thermostat, and Google Chromecast.

To allow the blue and red teams to function separately and work as opponents, only one team had access to the lab space at a time. The blue team would have time in the lab to create defenses and then the red team would have time to perform attacks. Our team documented both the attacks that we were able to carry out and attacks that had the potential to be successful, but that we were unable or unwilling to perform for a variety of reasons. For more information on these limitations, see Section 6.

III. Activities

A number of different attacks were performed with a variety of different tools in an attempt to find the vulnerabilities in different IoT devices. This section will discuss some of those attacks in detail and the effects that they can have.

A. Network Spying Using Airodump-ng and Airgraph-ng of Aircrack-ng Suite

The more information about the target that can be gathered, the greater the chance there is of an attack succeeding and a vulnerability being exploited. Therefore, we began by researching and performing espionage on the lab environment. However, without access to the network, information about the traffic going between devices cannot be collected and learned from. One way to circumvent this issue is to sniff the WiFi radio traffic and learn about the relationships between the different devices on a network. Tools like airodump-ng, which is used for packet capturing, helps to sniff this kind of traffic. Unfortunately, the data collected by airodump-ng is not in a human-friendly format. It essentially draws a map of the relationships between the devices in a binary format. To be able to convert this data into a graphical, human-readable format, other tools like airgraph-ng were used. Airgraph-ng can be used to produce two different kinds of graphs: graphs of access point relationships and graphs of probe frames.

Two types of graphs were used to analyze the network. Graphs of access point relationships (GAPR) are more commonly known as connected devices graphs. They map access points and all of the devices that are connected to those access points. Graphs of probe frames (CPG) are also called disconnected devices graphs. Those graphs show devices that are not connected to any access points and all of the networks that those devices have ever been connected to. To create either GAPR or GPU graphs, a WiFi adapter that supports monitor mode and that has an aircrack-ng suite installed are required. Monitor mode is a data capture mode that allows for the collection and monitoring of data that is received on a wireless channel. Airgraph-ng, airodump-ng, and aireplay-ng are all part of aircrack-ng, which is a complete suite of tools that assess WiFi network security. Aircrack-ng can be used to monitor packet capture and export the data to text files for further processing, perform replay attacks, de-authentication, fake access points, and more via packet injection, and crack WEP and both WPA1 and WPA2 encryption. In this experiment, we used the Alfa AWUS036ACH WiFi adapter and used the airodump-ng and airgraph-ng script that is included in aircrack-ng.

The procedure to perform network spying with these tools is as follows:

1. Set WiFi adapter to monitor mode.

    $ ifconfig wlan0 down

    $ iwconfig wlan0 mode monitor

    $ ifconfig wlan0 up

2. Capture surrounding network traffic using Airodump-ng.

    $ airodump-ng wlan0 -w IoT-Lab-captured-traffic

3. Generate a graph of devices that are connected to the access point (CAPR Graph).

    $ airgraph-ng -i '/root/Desktop/IoT-Lab-captured-traffic-01.csv' -o IoT-CAPR.png -g CAPR

4. Generate Graph of devices that are not connected to the access point (CPG Graph).

    $ airgraph-ng -i '/root/Desktop/IoT-Lab-captured-traffic-01.csv' -o IoT-CPG.png -g CPG

This procedure creates both a CAPR graph and a CPG graph. Figure 1 shows the result produced from step 3, which is the connected devices (CAPR) graph of our smart home lab. It shows all devices that are connected to our smart home lab network via the OpenWrt router, with all of their MAC addresses and extra details about the devices if possible. Those MAC addresses are blacked out on the graph in Figure 1 to protect our privacy.

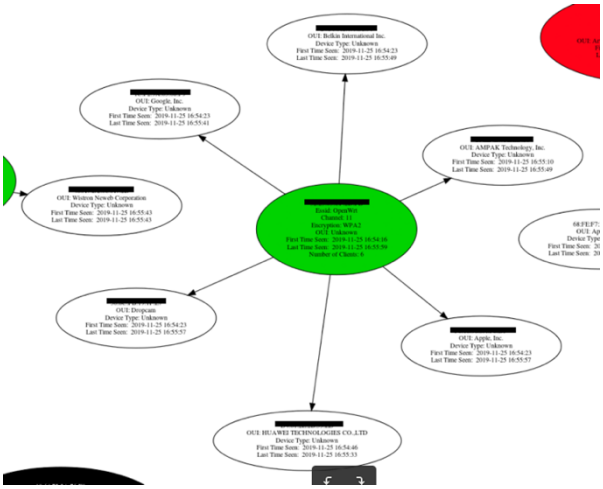

Figure 1: CAPR Graph of Smart Home Lab

Figure 2 shows a zoomed-in part of the graph from Figure 1. In this part of the graph, airgraph-ng can inform us that there is a Dropcam that is connected to the router called OpenWrt, which is the smart home lab router, and it's MAC address is on channel 11. It also shows the MAC address of the Dropcam itself. All of this information could potentially be very valuable for hackers. Attackers could use this information to do a lot of damage to these devices. Section 4.2 shows in detail one example of how this information could be used to perform an attack.

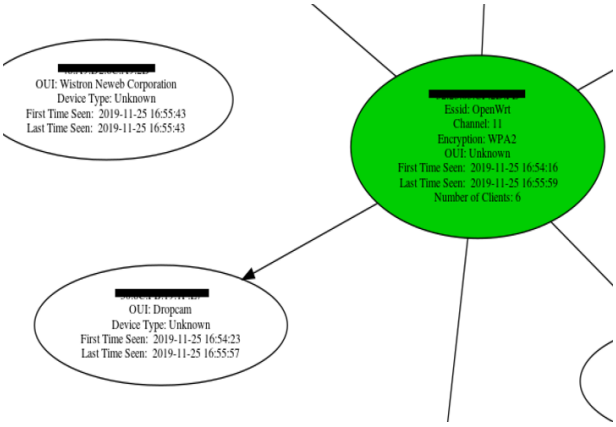

Figure 2: CAPR Graph Dropcam Information

Figure 3 shows an example of the disconnected devices (CPG) graph of our smart home lab. This graph shows all of the devices with their respective MAC addresses that surround the smart home lab, including those that are connected to the lab's router and those that are not. It also shows the name of the networks that all those devices had in the past been connected to. Again, like with Figure 1, the MAC addresses of the devices are blacked out to protect our privacy. Attackers could take advantage of this information by creating fake mockups of those networks, tricking users to login, and then stealing their credentials. We did not perform such attacks in this experiment because we wanted to focus on attacks that target technical flaws instead of social engineering attacks.

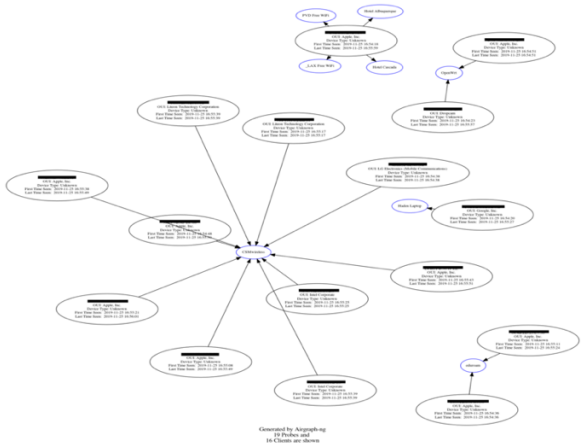

Figure 3: CPU Graph of Smart Home Lab

Zooming in on the CPU graph, as can be seen in Figure 4, it can be seen that there is an Apple product, probably an iPhone, that is not currently connected to the smart home lab router, OpenWrt. But that device has been connected to four different networks: PVD Free WiFi, Hotel Albuquerque, _LAX Free WiFi, and Hotel Cascada. This is a clear red flag for protecting users' privacy.

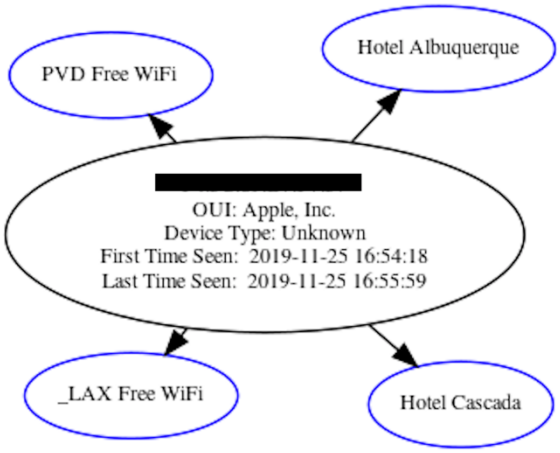

Figure 4: CPU Graph Apple Product Information

B. Disable Dropcam Security Camera Using De-Authentication Attack

The information provided by airgraph-ng can be used to perform a simple attack to disable a security camera. In our experiments, a Dropcam IoT security camera was the attack target. A de-authentication attack using the aireplay-ng script included in the aircrack-ng suite can be used to perform this attack. All that is required is a WiFi adapter that supports monitor mode and a PC with the aircrack-ng suite installed, just as is required to perform network spying.

The procedure to disable a Dropcam with these tools is as follows:

1. Set WiFi adapter to monitor mode.

    $ ifconfig wlan0 down

    $ iwconfig wlan0 mode monitor

    $ ifconfig wlan0 up

2. Identify the client device and its access point. In this particular experiment, the client is the Dropcam with its MAC address and its access point is the smart home lab router on channel 11 with its MAC address.

3. Set WiFi adapter to the channel the two devices are on.

    $ airmon-ng start wlan0 11

4. Perform the attack using the following command.

    $ aireplay-ng -0 0 -a aa:aa:aa:aa:aa:aa -c xx:xx:xx:xx:xx:xx wlan0

When the command is executed, it will jam the WiFi connection between the router and the Dropcam by continuously sending authentication packets masquerading as a router with MAC address aa:aa:aa:aa:aa:aa. The results of this attack can be seen in Figures 5 and 6. Figure 5 shows the output from the Dropcam before the attack and Figure 6 shows the output after the attack.

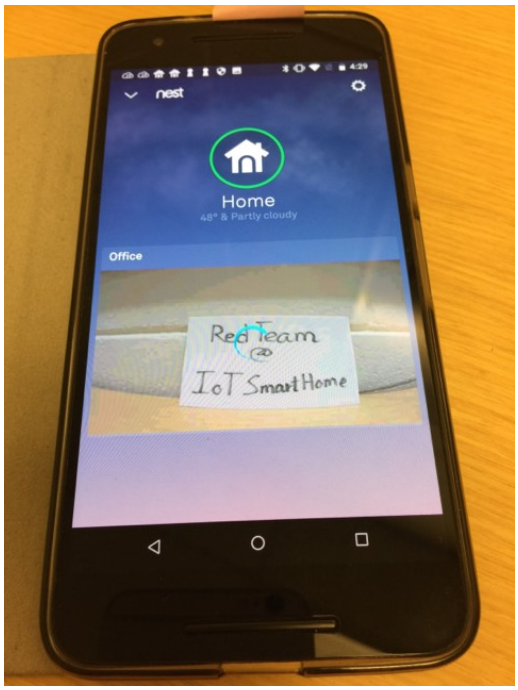

Figure 5: Dropcam Output Before Attack

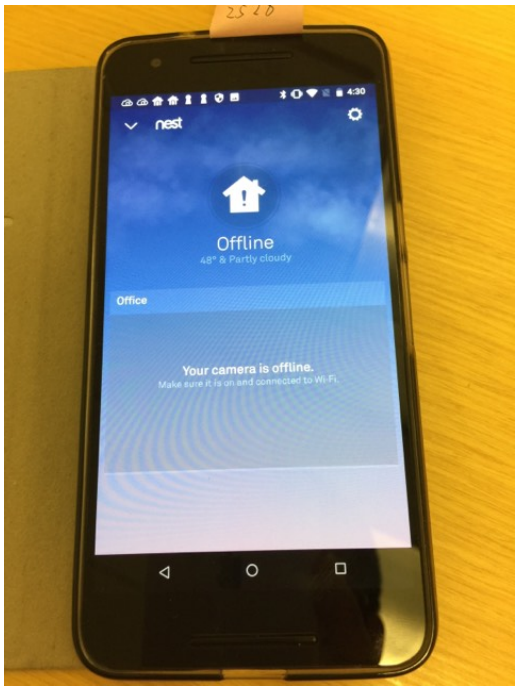

Figure 6: Dropcam Output After Attack

C. Disable Wemo Smart Plug Using De-Authentication Attack

In this attack, the de-authentication technique was also used to disable the client device from the access device. Assuming that all of the necessary data is gathered, we can apply this same procedure to attack the Wemo smart plug. The only difference is that the target MAC address has to be changed to the Wemo's address. Figure 7 shows the Wemo status before our attack is performed. At this moment, the Wemo smart plug is active and the user can use the Wemo app on a smartphone to control the smart plug. Figure 8 shows the status of the smart plug after being attacked. The smart plug has been disconnected from the control app and the user has completely lost control of the Wemo smart plug.

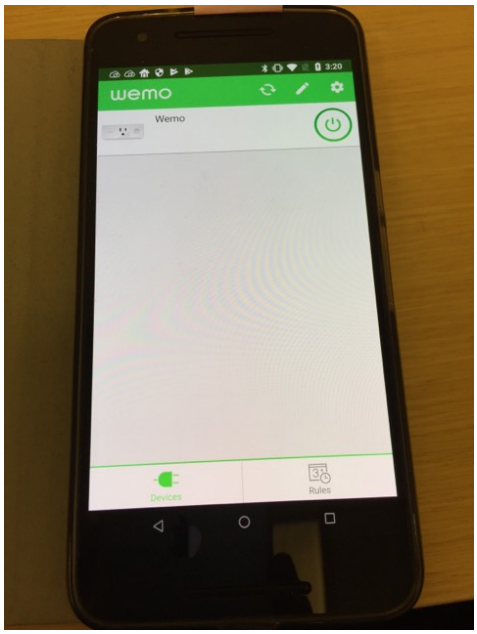

Figure 7: Wemo Smart Plug Before Attack

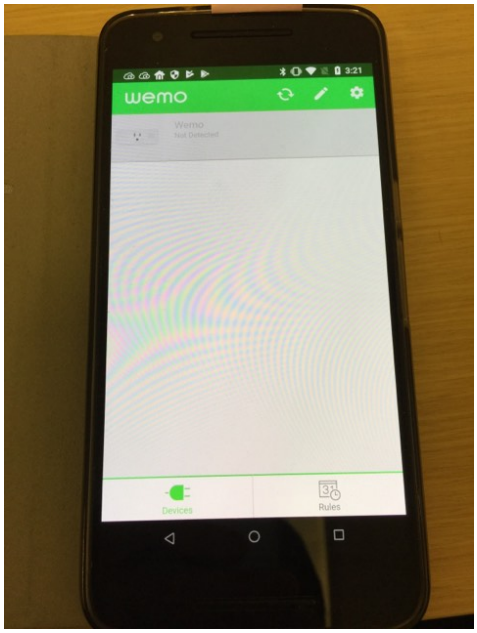

Figure 8: Wemo Smart Plug After Attack

D. *Disconnect Google Chromecast Using De-Authentication Attack*

In this attack, we again used the de-authentication technique to disconnect the Google Chromecast in our smart home lab. The Google Chromecast device used in this experiment is the newest, third-generation Google Chromecast. The procedure is entirely the same as with the Dropcam and the Wemo plug. Again, the only thing we need to do is set the target MAC address to the Chromecast's address. Figure 9 shows a YouTube video being cast to the Google Chromecast. Figure 10 and 11 show the result of our attack; we can see that the Google Chromecast device is disconnected from the network and the video is no longer being cast. Figure 11 shows that the Google Chromecast is totally offline, which blocks users from casting any new videos and it affects the usability and availability of this device.

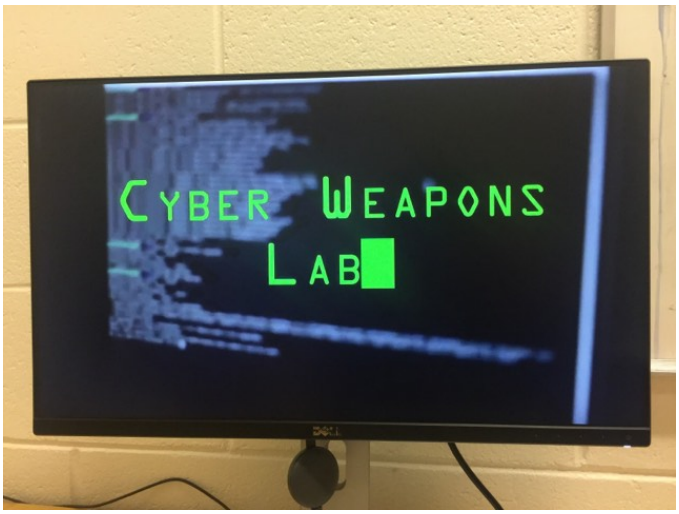

Figure 9: Google Chromecast Before Attack

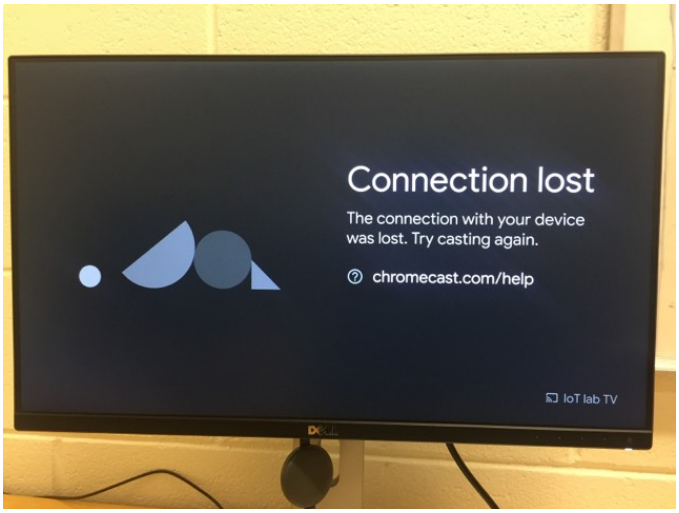

Figure 10: Google Chromecast First Response After Being Attacked

Figure 11: Google Chromecast After Attack

E. *WPS Attack*

Another attack that can be performed is a WPS attack. This attack exploits a router convenience feature to gain access to the network. WPS stands for WiFi Protected Setting; it is a feature that allows individuals to use an 8-digit WPS pin number that is printed on the router to login to the network. To perform this attack, a WiFi adapter is again required that supports monitor mode. The WiFi adapter must also work with the Airgeddon script. Airgeddon is a multi-use bash script used to audit wireless networks. It can be used to perform attacks such as Pixie Dust attacks, Brute Force Pin attacks, WPS attacks, and many more. The Alfa AWUS036NHA WiFi adapter was used in this attack. For this attack, the WiFi adapter had to be changed to Alfa AWUS036NHA instead of Alfa AWUS036ACH, because the chipset of Alfa AWUS036ACH is a newly Linux supported chipset, so its driver has not been developed to be compatible with Airgeddon yet.

To perform this attack, the first step is to start the Airgeddon script, select the WiFi adapter, and set it to be in monitor mode. Next, the target needs to be found and set, which in this case is a router that has the WPS feature enabled. Then, the attack module is chosen, and the attack is performed. Because Airgeddon has many different attack modules, the attacker needs to explicitly choose which module they want to use. If the attack succeeds, it will output the username and the password to the network. We were able to successfully perform this attack on a D-Link router on the home network of one of our team members. However, it did not work with our smart home lab router because the smart home lab router uses the OpenWrt operating system that disables the WPS feature by default. Despite this, WPS is a durable attack on many different kinds of routers. Because of this, we need to pay attention to our WPS settings because if someone can get access to your network, they can do a lot more damage to the devices within your network than was previously believed. In the next section, another attack is described in detail which shows just how simple it is to hijack any Google Chromecast device on a network.

F. *Google Chromecast Hijacking*

The common weakness of IoT devices is that they normally don't have their own security layer. In most IoT devices, their security relies on the network security layer. This is a very bad security design approach. Therefore, attacking devices such as Google Chromecast becomes super easy when attacker get access to the network. In this attack, we hijacked a Google Chromecast at our smart home lab using the Cast All The Things (CATT) script. Thanks to tools that have been developed specifically for attacking Google Chromecast devices, such as CATT, hijacking this kind of device is very easy. CATT is a free to use script that allows attackers to hijack Google Chromecasts and display a variety of different media that the user or owner of the Chromecast has not permitted to be shown. Any operating system that has Python installed upon it can be used to run a CATT script.

To hijack a Google Chromecast using the CATT script, the general procedure is as follows:

1. Scan the network to find Chromecast devices

    $ catt scan

2. Start casting thing to the device

2.1 Cast any online video:

    $ catt cast <link to the online video>

2.2 Cast any website:

    $ catt cast_site <link to the site>

2.3 Cast a local video

    $ catt cast <path to the video file>

2.4 Cast a local video with added subtitles:

    $ catt cast -s <path to subtitle file (.srt)> <path to the video file>

Note that in case the network has multiple Google Chromecast devices we can use -d option to specify a device that we want to cast to.

```
[redacted]$ catt scan
Scanning Chromecasts...
[redacted] - Bedroom - Google Inc. Chromecast
[redacted] - LivingRoom - Google Inc. Chromecast
[redacted]$
```

Figure 12: CATT Screenshot

In Figure 12, we can see that the network has two Google Chromecast devices that were found along with their IP addresses: Bedroom and LivingRoom. The sample command below will only cast things to the LivingRoom Google Chromecast device:

    $ catt -d LivingRoom cast <source video>

While innocent material could be cast to a Chromecast device, malicious material could just as easily be displayed. Harmful or sensitive videos could be cast to the device. In addition, the subtitle functionality could be used to send or display messages. Videos and messages could be used as a means of propaganda or to harm in some fashion. However, innocent or not, the user or owner of the Chromecast no longer has control over their device, and this violates the availability component of the CIA triad.

IV. Analysis

The default username and password for a router can easily be found and exploited by attackers. It is therefore important for users to alter their router's username and password to something more secure. It is also important for users to be aware of what kind of devices are connecting to their network and in what ways those devices are connecting. For example, when the Dropcam was connected to the network, its title was simply "dropcam". This could easily be exploited by viewing the network. Once they know that the user is using a Dropcam, they can use a Dropcam-specific attack and the attackers are guaranteed to be able to disable the camera. However, this could also be used to the user's advantage. For example, changing the name of a Dropcam to Amcrest Pro HD, a rival camera, might convince attacks to use the wrong kind of attack to disable the camera.

It is clear from the results found that IoT devices push usability and convenience over security. The blue team consisted of students who were familiar with tech, not the average user. The average user doesn't change default passwords and so this is who are the experiments were conducted against. The blue team, however, deployed some security measures after we conducted our attacks to help resolve the issues we found. This consisted of turning off the WPS feature on the router, making their IPs static, whitelisting their needed MAC addresses, segmenting their network, installing a VPN, and setting up an intrusion detection system (IDS) with Snort. If we were to attempt to conduct our attacks again, we would have to change the way we attacked their system. Due to the fact they disabled WPS on their router, we would have to find a different way to gain access to the router and their network. If we were unable to gain access, our attacks on the IoT devices would consist of disabling rather than gaining control. For example, our attacks on their Chromecast device would not be possible because we weren't inside the network. We could instead create a de-authentication attack on the Chromecast in hopes of disabling its features. The security features implemented by the blue team did a good job of warding off our attacks. It would be great if the average user could employ these security features, but that is asking a lot considering they require a decent amount of technical expertise. At the very least, average users should make sure to change default passwords, disable WPS on their router, and keep an eye on their network device names to ensure they don't give away too much information.

Our lab setup currently has 4 WiFi devices that are being using: Foscam R4, indoor DropCam, Wemo smart plug, and Google Chromecast. We are able to disable 3 of them using de-authentication technique. This clearly shows that WiFi

devices are very breakable. The disabling of these devices could cause serious damage to the system in many different ways. For example, by disabling the WiFi cameras in a house, an attacker can bypass the physical security of the house. Losing control of a smart plug could cause serious damage to other devices that use the smart plug as their power source. Homeowner's health could be affected if they lost control of their smart plug and could not turn on the heater during an extremely cold day.

V. Limitations

The biggest challenge and limitation for this experiment was the fact that the router used was connected to the Colorado School of Mines network. The network setup was required to follow the school's network security policies, and this caused a number of different issues. Initially, there were technical difficulties with getting the router up and running, which caused delays in getting started. Beyond that, there are also a number of restrictions as to what can be done on the Mines network. The Mines network is much more protected and secure than an average home environment where many IoT devices can be found. Many tools and attacks that we wanted to use could have been performed on a normal home network but were impossible on the Mines network. For example, the IT department (ITS) restricts certain types of traffic like ICMP when it comes from an internal router. In addition, since our router was connected to the Mines network, we were hesitant and at times unwilling to perform certain kinds of attacks that had the potential to go beyond the lab and into the overall network.

Another big challenge had to do with scheduling. Initially, we planned to alternate each team's access to the lab every week. However, splitting up the lab time between the red and blue teams like this made it difficult for each team to be able to accomplish their goals in a timely manner. Therefore, about halfway through the experiment, it was decided that the lab space and time would be shared and either team could work at the lab whenever they wanted or needed to. This greatly increased productivity, but at the cost of privacy and secrecy. By sharing the lab time, both teams were often working at the same time and therefore each team knew what the other team was working on. Therefore, the red team often knew what vulnerabilities the blue team had not yet secured against and the blue team often knew what attacks the red team was planning. While in some ways this speed up the experiment as the teams were cooperating, it also might have added a certain level of dependence to the experiment. Since the red team was focused on exploiting the vulnerabilities that we knew had not been taken care of, it is possible that we focused less on coming up with new and unusual ideas that were not related to what the blue team was doing.

VI. Future Work

In the future, we hope to work with more of the provided devices to help gain an understanding of how our current attack strategies could be used on them. We were given a total of 13 devices but were only able to perform attacks on four of them in the allotted time. A large portion of the remaining nine devices were IoT devices, so the de-authentication attacks we performed could be performed to disable or even gain full control of the devices. We are currently looking into deploying the Mirai botnet on our IoT devices to monitor how it spreads and affects the devices.

The Mirai botnet source code is publicly available on GitHub along with step by step instructions to set it up. The problem is these instructions only show how to set up the fully-fledged botnet which would probably break out of the lab and attack other devices if we tried to run it in our environment. Instead of risking this, we decided it would be safer to modify the source code and only run the brute-force attack on our devices. This would prove that these devices are vulnerable to Mirai without risking trouble with the IT department and/or the law. The source code is poorly documented, and we were unable to figure out how to modify it given our time restrictions. We are also concerned that the SYN scan used by Mirai to find ports to attack will be blocked by ITS. Despite this, Mirai will be a good attack to try to perform in future work.

Furthermore, we only focused on technical vulnerabilities of IoT devices and the blue team consisted of all tech savvy people. No social engineering attacks were performed, but social engineering attacks are very powerful, and they normally bypass security layers completely. In the future, we would like to demonstrate a number of different social engineering type attacks so that users can be aware of those attacks in their daily lives so that they can better protect themselves.

VII. Conclusion

"Internet of things" is a common phrase in today's world. The internet of things, or IoT, comprises of a series of electronic devices that are connected together and to the internet. They allow unprecedented data collection of our world. These devices bring many advantages to our everyday lives, but they also come with many security concerns, especially in terms of privacy. Currently, if a user chooses to use an IoT device then they must also accept all the risks that come with the convenience it provides. The risks associated with IoT devices are many and vast, but there are very few security measures in place to help circumvent those risks. To help showcase just how dangerous these devices can be, this paper presents some attacks performed by novices to find and highlight the vulnerabilities in smart home systems. This work attempts to provide awareness of these vulnerabilities in the hope that in the future, better security measures will be implemented that will protect the privacy of users.

References

[1] Z. Zhang, M. C. Y. Cho, C. Wang, C. Hsu, C. Chen and S. Shieh, "IoT Security: Ongoing Challenges and Research Opportunities," *2014 IEEE 7th International Conference on Service-Oriented Computing and Applications*, Matsue, 2014, pp. 230-234.

[2] R. Mahmoud, T. Yousuf, F. Aloul and I. Zualkernan, "Internet of things (IoT) security: Current status, challenges and prospective measures,"

2015 10th International Conference for Internet Technology and Secured Transactions (ICITST), London, 2015, pp. 336-341.

[3] C. Lee, L. Zappaterra, Kwanghee Choi and Hyeong-Ah Choi, "Securing smart home: Technologies, security challenges, and security requirements," *2014 IEEE Conference on Communications and Network Security*, San Francisco, CA, 2014, pp. 67-72.

[4] A. Dorri, S. S. Kanhere, R. Jurdak and P. Gauravaram, "Blockchain for IoT security and privacy: The case study of a smart home," *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, Kona, HI, 2017, pp. 618-623.

[5] Kang, W.M., Moon, S.Y. & Park, J.H. An enhanced security framework for home appliances in smart home. *Hum. Cent. Comput. Inf. Sci.* 7, 6 (2017) doi:10.1186/s13673-017-0087-4

[6] Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J.A., Invernizzi, L., Kallitsis, M., Kumar, D., Lever, C., Ma, Z., Mason, J., Menscher, D., Seaman, C., Sullivan, N., Thomas, K., & Zhou, Y. (2017). Understanding the Mirai Botnet. *USENIX Security Symposium.*

[7] Mateti, Prabhaker (2005), Hacking Techniques in Wireless Networks: Forged Deauthentication, Department of Computer Science and Engineering, Wright State University.

[8] Bellardo, John; Savage, Stefan (2003-05-16), "802.11 Denial-of-Service Attacks: Real Vulnerabilities and Practical Solutions", Proceedings of the USENIX Security Symposium, Aug 2003