Vulnerabilities in Hub Architecture IoT Devices

Bogdan Alexandru Visan[1], Jiyoon Lee[2], Baijian Yang, Anthony H. Smith and Eric T. Matson
Computer and Information Technology
Purdue University
West Lafayette, Indiana, USA
{bvisan, lee1742, byang, ahsmith, ematson}@purdue.edu

Abstract— **This paper introduces new methods for gaining sensitive information from Internet of Things (IoT) devices. Generally, manufacturers tend to forget about security when putting a new product on the market, this notion does not exclude IoT. This has been proven by current research using simple techniques and devices to obtain unencrypted information. Using similar attacks, the IoT hubs themselves was chosen as a target instead of the devices connected to them. Two different attacks will be used. First, using various methods for sniffing Hub traffic, we will attempt to gain credentials on the uplink side. Secondly, port scans will be used to gain any exploited services in order to obtain root access. After our attempts, it was found that IoT hubs can be very susceptible to tech savvy attackers using simple Man in the Middle attacks. If network access was achieved, the security of ones home is greatly compromised allowing intruders an 8-10 minute window to enter a house undetected. Using modest methodologies, consumers and companies can prevent future exploits.**

I. INTRODUCTION

Internet of Things (IoT) is a novel paradigm that is continually and rapidly evolving into a modern wireless technology. This allows people to interact with things such as lamps, sensors and doors. IoT already plays many roles such as Intelligent Transportation System (ITS), business/process management, Wireless Sensor Networks (WSN), and assisted living [7].

The main idea of the IoT is to link things (or IoT devices) which people use on a daily basis, and enable us to manage household items remotely. By using the IoT technology, people can easily manage both private and work devices in a relatively secure environment. Obviously, as IoT devices become more prevalent, consumers are more likely to use this technology. In the perspective of individual users, there are many possible IoT applications such as assisted living, enhanced learning, and e-health. In the foreseeable future, these applications will play more important roles in the private area as well. This means IoT applications will be available in commercial fields such as automation, industrial manufacturing, logistics, and intelligent transportation for business users [4].

Since 2008, National Intelligence Council (NIC) considered IoT to be a part of Disruptive Civil Technologies because of its potential impact on US national power [5]. They said that in the future, Internet nodes reside in such everyday things as food packages, furniture, paper documents, and more. Moreover, they also predict that Popular demand combined with technology advances could drive widespread diffusion of an Internet of Things (IoT) that could, like the present Internet, contribute invaluably to our economy.

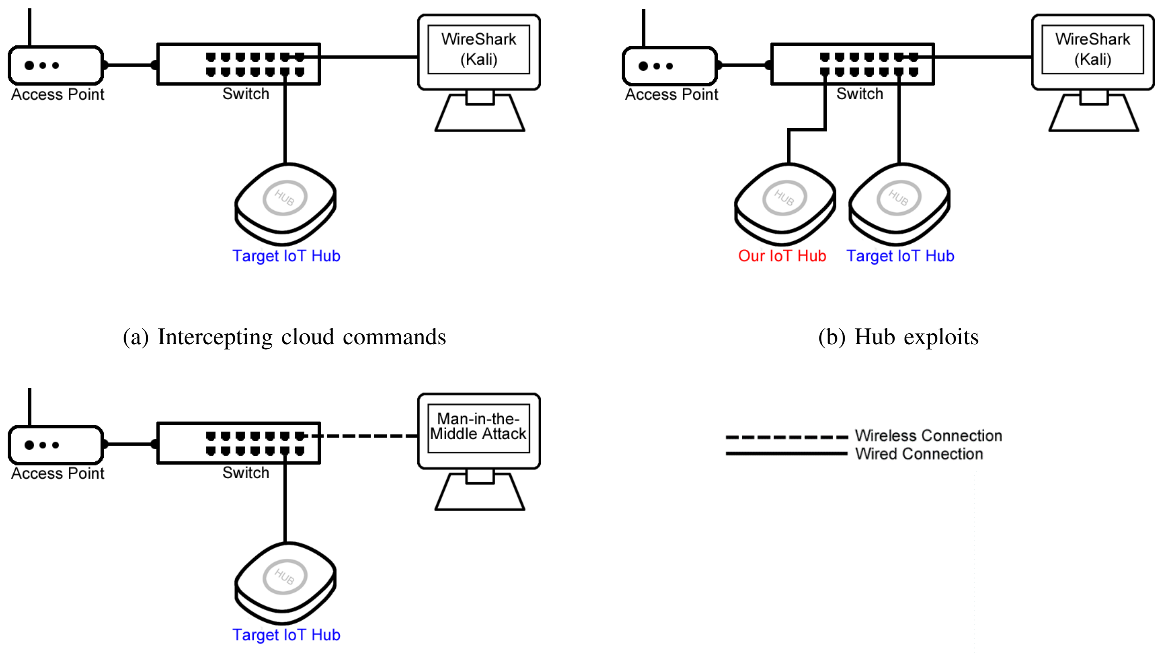
IoT devices are usually connected to a central hub that sends specific instructions based on the users desire. If the hub was compromised by an attacker, they could control all the devices on the network by sending wrong instructions. This means that the attacker can potentially control your sensors, devices, and even your home. For example, lets assume that a thief attacks your IoT system before breaking into your home or office. If they have access of it, the thief may control door locks and alarms making a clean escape easier. Moreover, if an IoT network is used in a medical environment, having unauthorized access could cause complications with patients. In NICs perspective, use of IoT devices will be rampant in the near future, and it directly affects our life. This is why securing the Internet of Things must be a priority.

This paper will introduce possible vulnerabilities in current IoT hubs and suggest novel solutions to any vulnerabilities found. The remainder of the paper consists of as follows. In Section 2, some previous works which have been done to find vulnerabilities of IoT systems and attack their connection will be presented. A detailed description of methodologies which was used to attack IoT system are given in Section 3. An experimental settings and results of experiments are subject of Sections 4 and 5. Conclusions and clues of future work are in the last section, Section 6.

II. RELATED WORK

IoT devices and control units use a wide range of wireless protocols in order to communicate with each other. Protocols such as ZigBee, Bluetooth, 6LowPAN, and Wi-Fi can all used. However, these standards have vulnerabilities which could expose a users private information to attackers.

ZigBee is one of the most popular standard used, as it supports low data rates wireless connectivity designed for short range low power devices [2]. Because of this, IoT devices tend to use the ZigBee standard for wireless communication. A drawback of ZigBee is the use of unencrypted communication to make it easier for consumers. This means an eavesdropper can capture packets and decode them using a cheap ZigBee sniffer, KillerBee, and WireShark. [11] Replay attacks are also possible as mentioned in [11] by sending the same message with a higher frame counter. This can be mitigated by using a timestamp along with the frame



(a) Intercepting cloud commands

(b) Hub exploits

(c) Wireless attack

(d) Type of line in figure

Fig. 1: Methodologies

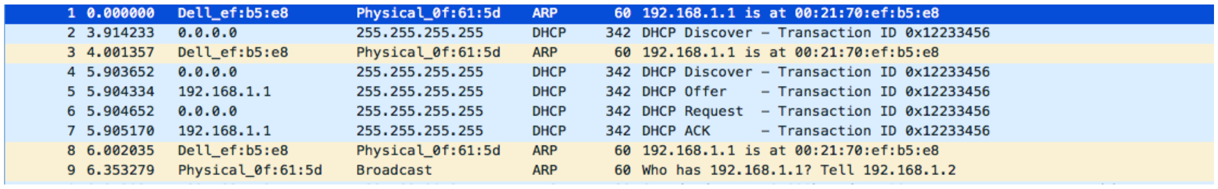
counter in the stack. [11] Radmand et al. also showed a survey of existing threats in ZigBee and introduced two different attacks; remote and physical to achieve the secret key in the ZigBee connection [12]. This can be mostly mitigated by encrypting the SKKE transactions [3]. In [6], Dini and Tiloca propose security concerns in the Smart Energy Profile. For example, one of the concerns relate to key management. When a device is suddenly disconnected, ZigBees network key and link key are still retained and may connect with peer devices. Dini and Tiloca also suggest a possible approach as the solution. Vidgren et al. introduce two practical attacks based on several vulnerabilities in the main security components of ZigBee [15]. The first attack is a sort of Denial of Service attack. By sending a special signal, they make the ZigBee device wake-up continually and eventually make the battery run out. They suggest another attack that is based on using the key exchange process in ZigBee as well. Recently, Zillner measured vulnerabilities in the ZigBee standard and ways to attack ZigBee devices [16]. The author points out that ZigBee communication is very easily jammed and an attacker can get complete control of system by enforcing a re-pair, thus sniffing the transmitted network key.

Man-In-The-Middle (MITM) attacks usually happen between two parties. The attacker creates independent connection with each party and makes the parties believe they are communicating with each other. Two different Man-In-The-Middle Bluetooth attacks were introduced by Haataja and Toivanen [8]. In the first methodology, they use a vulnerability throughout the IO exchange during the first phase of Secure Simple Pairing. That is possible because the exchange is done over an unsecured channel. In the second attack they persuade the victim to choose a less secure channel, but they need visual contact to the victims device. Furthermore, battery exhaustion Denial of Service attack (DoS) in Bluetooth can be used for IoT devices [10]. The attack is based on idea of that DoS attack can accelerate battery consuming.

IPv6 over Low Power Wireless Personal Area Networks, called 6LoWPAN, define a packet fragmentation mechanism to support the transmission of IPv6 packets which have a larger size than the maximum frame size. Hummen et al. describe two attacks using the aforementioned mechanism; the fragment duplication attack and the buffer reservation attack. They also suggest two lightweight defense strategies; the content chaining scheme and the split buffer approach [9]. Battery exhaustion attacks also directly affect 6LoWPANs lifetime and service availability [13]. Moreover, there are a number of routing attacks which are effective against 6LoW-PAN networks such as Wormhole attacks, Sinkhole/Black hole attacks, Selective forwarding, and Sybil attacks [13].

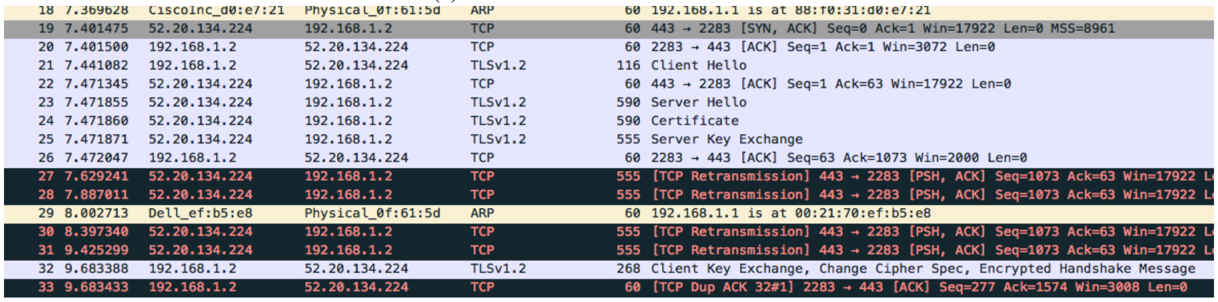
III. Methodology

This section describes two methodologies that were tested in a lab environment. For both scenarios, a preliminary test was conducted via a wired connection for proof of concept. If proven successful, the experiment would be tried via wireless connection. During wireless penetration testing, it will be assumed that access to the APs broadcasted network is achieved. This can be done via password cracking or



```
 1 0.000000   Dell_ef:b5:e8         Physical_0f:61:5d   ARP      60 192.168.1.1 is at 00:21:70:ef:b5:e8
 2 3.914233   0.0.0.0               255.255.255.255     DHCP    342 DHCP Discover - Transaction ID 0x12233456
 3 4.001357   Dell_ef:b5:e8         Physical_0f:61:5d   ARP      60 192.168.1.1 is at 00:21:70:ef:b5:e8
 4 5.903652   0.0.0.0               255.255.255.255     DHCP    342 DHCP Discover - Transaction ID 0x12233456
 5 5.904334   192.168.1.1           255.255.255.255     DHCP    342 DHCP Offer    - Transaction ID 0x12233456
 6 5.904652   0.0.0.0               255.255.255.255     DHCP    342 DHCP Request  - Transaction ID 0x12233456
 7 5.905170   192.168.1.1           255.255.255.255     DHCP    342 DHCP ACK      - Transaction ID 0x12233456
 8 6.002035   Dell_ef:b5:e8         Physical_0f:61:5d   ARP      60 192.168.1.1 is at 00:21:70:ef:b5:e8
 9 6.353279   Physical_0f:61:5d     Broadcast           ARP      60 Who has 192.168.1.1? Tell 192.168.1.2
```

(a) DHCP Lease Allocation Process



```
18 7.369628  CiscoInc_d0:e7:21  Physical_0f:61:5d  ARP        60 192.168.1.1 is at 88:f0:31:d0:e7:21
19 7.401475  52.20.134.224      192.168.1.2        TCP        60 443 → 2283 [SYN, ACK] Seq=0 Ack=1 Win=17922 Len=0 MSS=8961
20 7.401500  192.168.1.2        52.20.134.224      TCP        60 2283 → 443 [ACK] Seq=1 Ack=1 Win=3072 Len=0
21 7.441082  192.168.1.2        52.20.134.224      TLSv1.2   116 Client Hello
22 7.471345  52.20.134.224      192.168.1.2        TCP        60 443 → 2283 [ACK] Seq=1 Ack=63 Win=17922 Len=0
23 7.471855  52.20.134.224      192.168.1.2        TLSv1.2   590 Server Hello
24 7.471860  52.20.134.224      192.168.1.2        TLSv1.2   590 Certificate
25 7.471871  52.20.134.224      192.168.1.2        TLSv1.2   555 Server Key Exchange
26 7.472047  192.168.1.2        52.20.134.224      TCP        60 2283 → 443 [ACK] Seq=63 Ack=1073 Win=2000 Len=0
27 7.629241  52.20.134.224      192.168.1.2        TCP       555 [TCP Retransmission] 443 → 2283 [PSH, ACK] Seq=1073 Ack=63 Win=17922 L
28 7.887011  52.20.134.224      192.168.1.2        TCP       555 [TCP Retransmission] 443 → 2283 [PSH, ACK] Seq=1073 Ack=63 Win=17922 L
29 8.002713  Dell_ef:b5:e8      Physical_0f:61:5d  ARP        60 192.168.1.1 is at 00:21:70:ef:b5:e8
30 8.397340  52.20.134.224      192.168.1.2        TCP       555 [TCP Retransmission] 443 → 2283 [PSH, ACK] Seq=1073 Ack=63 Win=17922 L
31 9.425299  52.20.134.224      192.168.1.2        TCP       555 [TCP Retransmission] 443 → 2283 [PSH, ACK] Seq=1073 Ack=63 Win=17922 L
32 9.683388  192.168.1.2        52.20.134.224      TLSv1.2   268 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
33 9.683433  192.168.1.2        52.20.134.224      TCP        60 [TCP Dup ACK 32#1] 2283 → 443 [ACK] Seq=277 Ack=1574 Win=3008 Len=0
```

(b) Client/Server Handshake

Fig. 2: DHCP Lease Allocation Process (a) and Client/Server Handshake (b)

social engineering. In [14], it was shown that from a simple *Wardriving* experiment, 29% of people used no or outdated security such as WEP. Stimpson et al. also shows how easy it is to gain access to wireless networks using simple and free tools. Our methods were chosen due to the innate physical connection that smart hubs use to transfer sensitive information.

The required hardware and software used for all experiments include

- Cisco 1900 router appliance
- Cisco Catalyst 3650 PoE switch
- Cisco Aironet 1252AG AP
- Samsung SmartThings Hub x2
- Wireshark
- Kali (Metasploit, NMap)

Other configurations include having internet access. The Cisco 1900 was configured with a DHCP server that leases IP and DNS information. The Hubs network connection cannot be manually configured locally and must be automatic before working correctly.

A. Intercepting Cloud Commands

Our first test used the Samsung SmartThings Hub paired with a few devices such as a motion sensor, door sensor, etc. The specific devices paired wont be as important in the preliminary tests. The Hub was connected to a port on the Cisco 3650 that was then mirrored for monitoring. A computer capable of using Wireshark was plugged into the mirrored port in order to sniff traffic being sent and received by the Hub. This is shown in Figure 1.a.

While monitoring traffic, a user issued commands to the Hub via smartphone from a controlled account. The packets captured were analyzed for any decipherable information. We expected to gain sensitive information such as credentials, keys, or specific commands. This information could be used to log into the original cloud account, taking control of the Hub and its connected devices.

B. Hub Exploits

The second test was performed in a similar environment to Figure 1.b. A SmartThings Hub was connected to the Cisco 3650 with basic devices paired to it. We tried to use the tools that Kali provides in order to find a back door into the device. Being a low power IoT device, a Linux based distribution is likely to run on the Hub. A computer running Kali was connected to the same Cisco 3650. NMap was used to find any vulnerable service ports. Using this information, Metasploit can be applied to exploit any vulnerabilities.

Once root access was achieved, the dd command can be used to clone the firmware bit by bit to a second Samsung SmartThings Hub. This would cause both Hubs to have the same unique ID, which can be used to hijack the paired devices. In the event that copying the firmware would not work, root access could be achievable and might be used to gain sensitive information or manually send commands.

C. Wireless Attacks

Since attackers are most likely not going to have physical access to devices, our tests included a wireless Man-In-The-Middle (MITM) attack with the same above-mentioned experiments. Our experiment configuration is shown in Figure 1.c. Since the Hub relies heavily on internet access using DCHP, we would attempt to trick the device using ARP cache poisoning.

Assuming the target network has a wireless network and we were able to connect to it, Kali would be used to conduct the MITM attack. Using the *arpspoof* command, we could attempt to sniff the Hubs traffic. This would allow us to retrieve sensitive information while the user unsuspectingly goes about their day.

Fig. 3: Samsung SmartThings hub and sensors.

IV. EXPERIMENT

Samsung SmartThings first generation package was used to conduct experiments. The package contains SmartThings hub and various sensors such as motion sensor, outlet, arrival sensor, and multipurpose sensors (see Fig. 3.).

While little information about the inner workings of the Hub can be found online, using Wireshark can show us what is going on. As soon as the Hub is powered on and connected via Ethernet cable to a switch port, the DHCP Lease Allocation Process begins as shown in Fig. 2 (a).

If a DHCP server is not located on the network, the Hub will continually reboot, restarting the DHCP process. As a result of tests, we were not able to statically set an IP address to the Hub. Since the Hub is a cloud device, internet access along with DNS information must also be included in the DHCP lease.

Once the hub successfully has internet access, and is able to resolve one of many Amazon AWS Servers leased to Samsung, it proceeds to generate the session key between itself and the server. This step is required in order for the Hub to securely communicate with the Cloud and function normally. Fig. 2 (b) shows the Client/Server Handshake.

Looking at each packet carefully, it was observed that the Hub uses the Diffie-Hellman Key Exchange to generate session keys, RSA public key algorithm to verify certificates using an RSA key size of 1024, AES128 Encryption with Cipher Block Chaining, and lastly SHA1 data validation. The certificate used for all SmartThings Hubs are self-signed by Samsung and expire in 2025.

The normal operation of the hub consists of sending TCP Keepalive messages every 10 seconds repeatedly unless data is being sent instead. At 60 second intervals, Application Data is being transmitted for which is assumed to be status updates to the Cloud. If the Hub is physically unplugged or unable to reach the server, a notification will be sent to the user within 8-10 minutes after the time of disconnect. The functionality of the Hub is important to keep in mind, as this process was used to our advantage.

V. EXPERIMENT AND RESULT

A. *Forging Certificates (SSLsplit)*

Since the Hub, and other IoT Cloud devices rely on using certificates to communicate with the Cloud, the first trial of experiments were attempting to inject a known certificate. SSLsplit is a tool that can be installed on most Linux distros and generates self-signed certificates on the fly. In our environment, a laptop with Kali Linux installed was used. As a control, SSLsplit was tested on a victims laptop to see if certificates were being injected properly. Before SSLsplit can work, a known certificate and key was generated using the following commands.

- openssl genrsa -out ca.key 1024
- openssl req -new -x509 -days 1826 -key ca.key -out ca.cer

IP forwarding and NAT rules were enabled to redirect packets destined for the victims device. Port 80 & 443 were chosen since they were commonly used by the Hub and client computers. This makes our computer act like a transparent proxy.

- sysctl -w net.ipv4.ip_forward=1
- iptables -t nat -F
- iptables -t nat -A PREROUTING -p tcp –dport 80 -j REDIRECT –to-ports 8080
- iptables -t nat -A PREROUTING -p tcp –dport 443 -j REDIRECT –to-ports 8443

Using $arpspoof$, initializing a Man in the Middle attack is possible. The command is very easy to run, and anyone with network access can accomplish it.

- arpspoof t victims ip r gateways ip

Now the victims device is passing all traffic through our Kali machine and it makes running SSLsplit possible.

- ./sslsplit -D -l connections.log -j /tmp/sslsplit/ -S logdir/ -k ca.key -c ca.cer ssl 0.0.0.0 8443 tcp 0.0.0.0 8080

After running the attack, we confirmed that our forged certificate was replacing verified ones on the victims laptop. Going to sites like https://www.chase.com allowed us to log the connection and anything that was submitted. While this attack worked, the victim would certainly know something was wrong since the self-signed certificate was not trusted.

Following the aforementioned steps against the Hub, the attack proved unsuccessful at reading encrypted data. While we tried to copy as much information from Samsungs certificate into our own, the signature Samsungs Hub was expecting did not match. After 8-10 minutes during the attack, the Hub timed out since it could no longer communicate with the Cloud.

B. Removing Certificates (SSLstrip)

The Hub did not accept a forged certificate, so removing it all together was the next step. In order to do this, we used SSLstrip, a tool that can be installed on most Linux distros. For our experiments, we used the same Kali machine and tested it against a victims laptop.

Same as before, we used $arpspoof$ in order to achieve a Man in the Middle scenario.

- arpspoof t victims ip r gateways ip

The NAT rules are configured in order to act as a transparent proxy to the victims device. Make sure the following rule is added if not already there.

- iptables -t nat -A PREROUTING -p tcp –dport 80 -j REDIRECT –to-ports 8080

Lastly, the SSLstrip command was executed to listen in on the redirected port. This allows us to see any data that is captured.

- sslstrip l 8080

Unfortunately, this attack was difficult to perform on a laptop computer since many websites and browsers are fully aware they are being cheated. When attempting this attack on the Hub, we observed similar results as injecting certificates. The Hub times out after 8-10 minutes and alerts the user as expected. We could not read any Application Data.

C. Brute Forcing (Hydra)

In order to find open ports and potential exploits, the Metasploit framework along with Nmap included in Kalis Toolbox were used. Armitage is an easy to use GUI that can run Nmap and Metasploit. Using a terminal window, execute $armitage$. It was observed that Telnet port 23 was open on the Samsung Hub. Although the Hub does not support Telnet or raw socket access for users [1], this gave us an opportunity to run password cracking techniques using Hydra, another powerful tool included with Kali.

To use Hydra, simply execute $xhydra$ in a terminal window. This will bring up a GUI to easily input parameters. At this point, the IP address was chosen as the target, and a small password text file for testing proof of concept. As a result of several trials, each password tried was considered valid. This is because the Hub did not output any error response, which Hydra relies on determining validation of passwords accurately. The Hub would just prompt for another password whether or not it was correct. To fix this, changes were made in the source code of Hydra to accommodate for the error case. This is shown in Fig.4. After putting a test password into the original version of Hydra, the Hydra checks the first returned output and if it returns null, Hydra considers the password invalid. However, the Hub just shows the same sentence, Password: when the password is incorrect. Therefore, the checking mechanism should be changed to check if the first line of output contains Password:.

Running this attack did not prove ineffective, as a large password dictionary can be easily used to gain Telnet access. Since this would take a lot of time and clock cycles to complete, it would be more efficient to conduct Denial of Service attacks instead. Having ports open are major security concerns, thus Samsung should consider disabling Telnet unless necessary. Although, it is viable that the password is changed periodically to combat this type of attack. It would be very easy to modify once it is securely connected to the SmartThings server.

D. Denial of Service (hping3)

Having failed in obtaining data from the Hub, we decided that causing a major failure would still prove successful. Being a small device, flooding the Hub with TCP SYN messages seemed viable. In our experiments we were able to see results when connected via wired connection. But in the real world, an attacker would most likely be across the street.

The tool of choice was hping3, located in the Kali package. This program is able to send flood messages as fast as the attackers computer can handle. Once the IP of the hub was found, the following command was executed in a terminal window.

- hping3 -c 100000 -d 120 -S -w 64 -p 21 –flood –rand-source target ip

At the time of the attack, a Wireshark capture was being taken and the program was unable to process the amount of packets coming per second. This attack proved to be successful when conditions and processing power were correct. Although, using two wireless cards on different computers were still not enough to bring down the hub via wireless connection. We did notice latency in using the IoT devices, but nothing major.

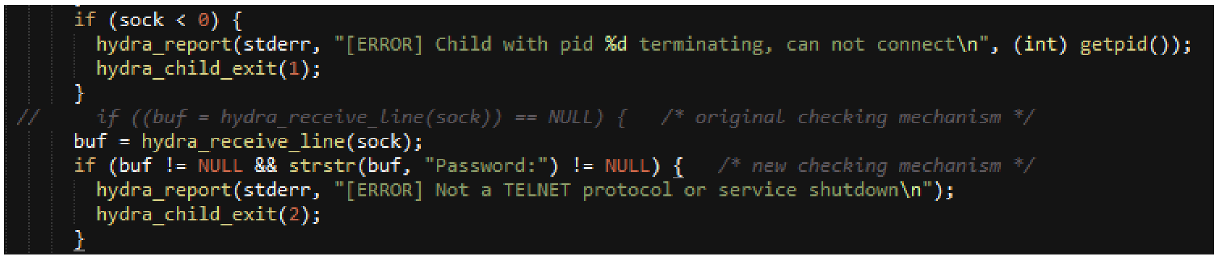
E. Denial of Service (iptables)

After the failure of the previous attacks, we took what we had and worked with it. In our Man in the Middle experiments, we noted that the Hub would time out and alert the user in around 8-10 minutes. It was also noted that all communications and alerts were not sending even though the Hub knew about the incident. This led us to find the simplest way an attacker can disrupt normal operations of the Hub. Simply black holing the Hubs traffic with $arpspoof$ and IP Forwarding disabled made the Hub lose connection to the Cloud. Once the attack began, the sensors connected to the Hub stopped responding to the mobile application, although they were still connected using 802.15.4.

- sysctl -w net.ipv4.ip_forward=0
- arpspoof t victims ip r gateways ip

VI. Conclusion and Future Work

From our experiments, we decided to focus on the security aspect of IoT Hubs as a whole. Because Samsung has one of the best Smart Home appliances, it is generally the most



```
    if (sock < 0) {
      hydra_report(stderr, "[ERROR] Child with pid %d terminating, can not connect\n", (int) getpid());
      hydra_child_exit(1);
    }
//      if ((buf = hydra_receive_line(sock)) == NULL) {   /* original checking mechanism */
    buf = hydra_receive_line(sock);
    if (buf != NULL && strstr(buf, "Password:") != NULL) {   /* new checking mechanism */
      hydra_report(stderr, "[ERROR] Not a TELNET protocol or service shutdown\n");
      hydra_child_exit(2);
    }
```

Fig. 4: Modified hydra-telnet.c code.

secure. In the lab environment, we primarily used a door and motion sensor. After arming the devices, the user would get a notification that an intrusion was detected if the sensors were disturbed. For example, a door opening, or movement in the home.

When connected to the wireless network, we black-holed the traffic on the Hub, thus losing the ability to contact the Cloud servers. When manipulating the sensors, the SmartThings mobile app showed all clear. After 8-10 minutes, Samsungs Servers notified the user that the Hub went offline. In a potential real world scenario, an attacker could infiltrate a homes wireless network, physically break in, and have an 8-minute window without tripping any alarm. Once the attackers left, the Hub would then notify the user of the intrusion after the fact. As more consumers start making their homes smarter, potential weaknesses in current protocols could prove worse than current home security systems.

After numerously attempting novel attacks, the Samsung SmartThings Hub proved to be quite robust. Although it has its pitfalls, we feel that relying on the Hubs home security features would be risky. Still one could argue that using a strong WPA2 home wireless password is the simplest solution. Without network access, an attacker would not be able to conduct the previous attacks mentioned above. Another viable defense strategy for stopping Man in the Middle attacks would have manufacturers include static ARP Tables. This may involve some user interaction during first time setup, but the possibility of having attackers spoof the Hub would be minimized greatly.

References

[1] SmartThings developer documentation SmartThings developer documentation.

[2] Standard: ZigBee PRO specification | the ZigBee alliance.

[3] Cristina Alcaraz and Javier Lopez. A security analysis for wireless sensor mesh networks in highly critical systems. 40(4):419–428.

[4] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. 54(15):2787–2805.

[5] N. Council. Six technologies with potential impacts on us interests out to 2025.

[6] Gianluca Dini and Marco Tiloca. Considerations on security in zigbee networks. In *Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC), 2010 IEEE International Conference on*, pages 58–65. IEEE.

[7] Daniel Giusto, Antonio Iera, Giacomo Morabito, and Luigi Atzori. *The Internet of Things: 20th Tyrrhenian Workshop on Digital Communications*. Springer Science & Business Media. Google-Books-ID: vUpiSRc0b7AC.

[8] Keijo Haataja and Pekka Toivanen. Two practical man-in-the-middle attacks on bluetooth secure simple pairing and countermeasures. 9(1):384–392.

[9] Ren Hummen, Jens Hiller, Hanno Wirtz, Martin Henze, Hossein Shafagh, and Klaus Wehrle. 6lowpan fragmentation attacks and mitigation mechanisms. In *Proceedings of the sixth ACM conference on Security and privacy in wireless and mobile networks*, pages 55–66. ACM.

[10] Benjamin R. Moyers, John P. Dunning, Randolph C. Marchany, and Joseph G. Tront. Effects of wi-fi and bluetooth battery exhaustion attacks on mobile devices. In *System Sciences (HICSS), 2010 43rd Hawaii International Conference on*, pages 1–9. IEEE.

[11] Olayemi Olawumi, Keijo Haataja, Mikko Asikainen, Niko Vidgren, and Pekka Toivanen. Three practical attacks against zigbee security: Attack scenario definitions, practical experiments, countermeasures, and lessons learned. In *Hybrid Intelligent Systems (HIS), 2014 14th International Conference on*, pages 199–206. IEEE.

[12] Pedram Radmand, Marc Domingo, Jaipal Singh, Joan Arnedo, Alex Talevski, Stig Petersen, and Simon Carlsen. ZigBee/ZigBee PRO security assessment based on compromised cryptographic keys. In *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2010 International Conference on*, pages 465–470. IEEE.

[13] Rabia Riaz, Ki-Hyung Kim, and H. Farooq Ahmed. Security analysis survey and framework design for ip connected lowpans. In *2009 International Symposium on Autonomous Decentralized Systems*, pages 1–6. IEEE.

[14] T. Stimpson, L. Liu, J. Zhang, R. Hill, W. Liu, and Y. Zhan. Assessment of security and vulnerability of home wireless networks. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2012 9th International Conference on*, pages 2133–2137, May 2012.

[15] Niko Vidgren, Keijo Haataja, Jose Luis Patino-Andres, Juan Jose Ramirez-Sanchis, and Pekka Toivanen. Security threats in ZigBee-enabled systems: vulnerability evaluation, practical experiments, countermeasures, and lessons learned. In *System Sciences (HICSS), 2013 46th Hawaii International Conference on*, pages 5132–5138. IEEE.

[16] Tobias Zillner and S. Strobl. *ZigBee ExploitedThe good, the bad and the ugly*. version.