Security Analysis on Consumer and Industrial IoT Devices

Jacob Wurm[1], Khoa Hoang[1], Orlando Arias[1], Ahmad-Reza Sadeghi[2] and Yier Jin[1]
[1]Department of Electrical and Computer Engineering, University of Central Florida
[2]Technische Universität Darmstadt, Germany

Abstract— The fast development of Internet of Things (IoT) and cyber-physical systems (CPS) has triggered a large demand of smart devices which are loaded with sensors collecting information from their surroundings, processing it and relaying it to remote locations for further analysis. The wide deployment of IoT devices and the pressure of time to market of device development have raised security and privacy concerns. In order to help better understand the security vulnerabilities of existing IoT devices and promote the development of low-cost IoT security methods, in this paper, we use both commercial and industrial IoT devices as examples from which the security of hardware, software, and networks are analyzed and backdoors are identified. A detailed security analysis procedure will be elaborated on a home automation system and a smart meter proving that security vulnerabilities are a common problem for most devices. Security solutions and mitigation methods will also be discussed to help IoT manufacturers secure their products.

I. Introduction

Within the past decade, the number of Internet of Things (IoT) devices being introduced in the market has increased drastically with the number of connected devices approaching 15 billion, and at roughly 2 devices per human this demonstrates a staggering conclusion that there are more connected systems than people living today [1]. This trend is expected to continue, with an estimate of 26 billion network connected devices by the year 2020, the majority of which being IoT and wearable devices [2]. Much like the embedded systems they derive from, IoT devices are armed with sensors but also offer some sort of connectivity functionality. As such, these devices can transmit the information they collect to a remote collection point.

The very nature of IoT devices is to collect, process and relay data through a communications channel and sometimes control a much larger unit. The data in question can range from a heartbeat, to the temperature of a room, to living habits and even the location of the user.

The ensuing privacy and security concerns that arise are known to manufacturers, however, security in IoT devices is either neglected or treated as an afterthought. Often, this is due to short time to market (TTM) and reduction of costs driving the device's design and development. As such, IoT devices remain open for attack and become attractive targets for those wishing to obtain the information they hold. Further, given the always-on network connectivity some of these devices hold and their different usage patterns, these devices could be targeted by malware, increasing the potential for harmful usage.

The few devices that do choose to add any protection usually employ software level solutions, such as firmware signing and the execution of signed binaries, methods which resemble those used in regular computing [3, 4, 5, 6, 7, 8, 9, 10, 11, 12]. These solutions, however, do not consider the different usage pattern that IoT devices have when compared to traditional embedded systems or personal computers. This proves to be insufficient, as not only have these mechanisms been proven to be bypassable but software-level protection schemes often leave the hardware unintentionally vulnerable, allowing for new attack vectors [13].

In order to better understand the security and privacy issues associated with the current IoT device design flow and their implications, we used two IoT devices as case studies. Given that IoT devices are widely used in both commercial and industrial applications, the selected sample IoT devices include a smart controller for a home automation system and a smart meter for modern power grids. Through our analysis, we will prove and demonstrate the limitations of current IoT device design methodologies when countering different cyber attacks from the hardware, software, and network levels. We will further develop countermeasures to mitigate security threats to existing IoT devices so that more secure devices can be deployed in the coming IoT era.

The remainder of the paper uses the following organization scheme. Section II introduces our security analysis of a sample commercial IoT device and reveals security vulnerabilities which exist in many commercial IoT devices. Section III presents our case study regarding a smart meter which is widely used in modern power grids. The analysis will show us that industrial IoT devices are equally vulnerable to cyber attacks. The security impact of the identified vulnerabilities and possible threat mitigating methods are discussed in Section IV. Conclusions are drawn in Section V.

II. Commercial IoT Device Security Analysis

Commercial IoT devices which directly target end-users are often designed with emphasis on device function-

ality. Security features are often added in an ad-hoc manner where remote attacks are treated as the main threats. Therefore, commercial IoT devices often suffer from hardware-level vulnerabilities [14] which may be remotely exploited. In order to demonstrate these security vulnerabilities and help designers/consumers better understand the design backdoors, the Haier SmartCare home automation system is selected as a case study in this paper.

A. High Level Overview

The Haier SmartCare is a smart device designed to control and read information from various sensors placed throughout a user's home which include a smoke detector, a water leakage sensor, a sensor to check whether doors are open or closed, and a remote power switch. These sensors are connected through the ZigBee protocol. The primary function of this device is to allow the user to better monitor their homes when they are away and to get alerts based on sensor information.

Fig. 1. Haier SmartCare Device (credit: Haier)

In order for users to connect to the device, they must first download a mobile application from the manufacturer's website. Next, they must connect the SmartCare to their network using an Ethernet connection. Following, they must connect their mobile device to the same local network as their SmartCare. Once it is connected, they must open the mobile application and create an account through the manufacturer's cloud service, which allows users to view their sensor data outside of their local network. Once this has been established, the users will be able interact with the sensors from their SmartCare through the mobile application.

B. Hardware Analysis

The first step in our vulnerability analysis was to analyze the components on the SmartCare's hardware platform. The main processing unit is a TI AM3352BZCZ60, which is a part of TI's Sitara line of processors. The processor contains an ARM Cortex A8 with NEON extensions. The processor also supports the use of operating systems such as Linux and Android. Upon analyzing the data sheet for the processor, we were able to locate traces for UART on the device. The SmartCare PCB is shown in Figure 2.

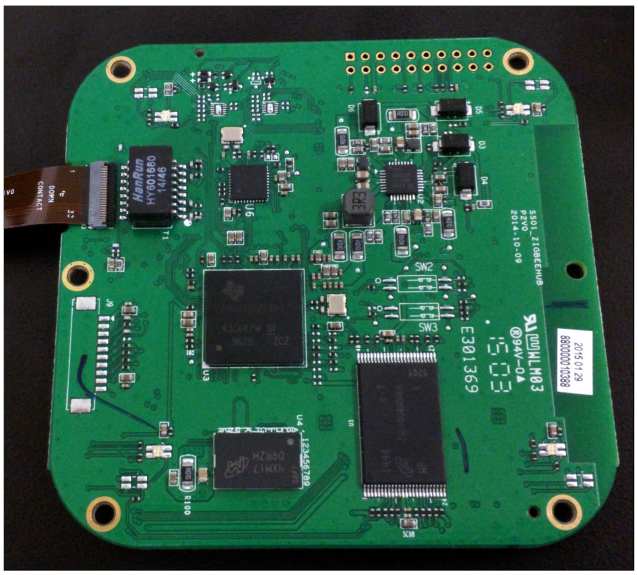

Fig. 2. SmartCare Hardware Platform

By leveraging the UART connection, we are able to read serial data from the device. By setting the correct parameters in the terminal emulator and connecting a serial-to-USB device to the SmartCare, we were able to view its start up sequence. In the beginning of the boot process, the device prompted us as to whether we would like to stop the automatic boot sequence. Upon stopping the process we were dropped into a U-Boot shell. It is here where the we were able to modify specific boot parameters for the device, such as where to start reading from memory, and what the initial shell will be. By modifying the initial shell among other variables, attackers will be able to gain low-level access to the device. After modifying the parameters we initiated the boot process. Once the device had finished booting up, we were dropped into a rudimentary shell.

C. Into the Shell

After reading the boot output of the device, it was apparent that this device was running Linux. Being on a Linux device, it is necessary to know what kind of permissions we have, running `id` showed us that we were on the root account of the device. Looking through the BusyBox utility showed us that the device is capable of

running a `telnet` server, allows for TFTP file transfer, and is able to fetch files from the web through `wget`.

Being on the root shell of the device also gave us the opportunity to look at the password hashes on the device, shown in Figure 3.

```
root@am335x-evm:/etc# cat /etc/shadow
root:RE0tDm4UmFgUo:16674:0:99999:7:::
```

Fig. 3. SmartCare Hashed Root Password

By referencing documentation on Linux shadow file structures, we were able to deduce that this device was using DES encryption on the password while also not using a salt. This means that the password is truncated to a maximum of eight characters then hashed. In order to obtain the root password for the device, the root password hash had to be cracked. The first attempt at cracking utilized a dictionary attack. In a dictionary attack, each password in the dictionary is hashed and subsequently checked against the hash in question. If the hashes match, then the password has been found, otherwise it will continue to check and hash each password in the list until it has reached the end. In this attack, a large word list containing approximately 32 million passwords was checked against.

Though 32 million passwords were checked against, none of them matched the root password of this device. The next option was a brute force attack, where every possible combination of characters is checked and hashed in order to find the root password. The total keyspace for a DES password using printable ASCII characters is $\sum_{i=0}^{8} 95^i$. This is a somewhat large keyspace, and may take hours or even days to go through every iteration on high performance hardware. Given that this method of attack is much more computationally intensive, we tried to optimize the cracking procedure leveraging high performance hardware with parallel processing capabilities. In our case study, we used two AMD R9 290 graphics cards to speed up the process

In our run, it took around five hours to get the root password. Since the root password for the device was known, the next course of action was to move onto another layer of attack. That is, we wanted to find out how we could attack other SmartCare devices using the secret learned from the device.

D. SmartCare Network Analysis

The new attack we tried to perform was a network-based remote attack. The first step in performing the network analysis was to scan the ports on the SmartCare to see if it is listening or transmitting on any of them. By performing a network scan we were able identify that the device may have had a `telnet` server running. Connecting to the device over telnet, we encountered a login prompt. Using the root credentials that were found earlier, we were able to get a root shell, which is shown in Figure 4.

```
        Haier



am335x-evm login: root
Password:
root@am335x-evm:~# id
uid=0(root) gid=0(root) groups=0(root)
root@am335x-evm:~#
```

Fig. 4. SmartCare `Telnet` Login Prompt

Since we were able to get a root shell over a local network, the next step was to see what kind of traffic this device generates. In order to analyze its network traffic, we had to perform a man-in-the-middle attack. This involved us using our computer as the gateway for the network the SmartCare was on. Through the gateway we were able to provide internet access. Using a packet sniffing program we were able to see what kind of traffic the device generates.

Once the network was up and running, we started the packet sniffer and looked at the network traffic. While most of the traffic going to and coming from the server was encrypted at the beginning, the device later fetched a firmware update over a plaintext HTTP connection, which is shown in Figure 5.

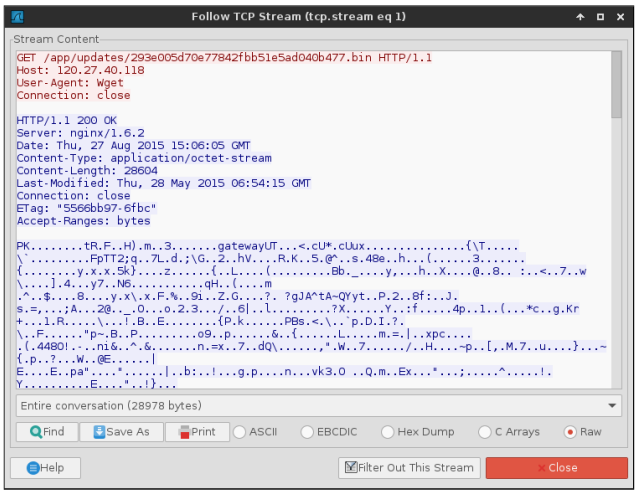

Fig. 5. SmartCare Fetching Update from Manufacturer's Server

As we can see in Figure 5, the first line in red indicates the package it wants to receive, which in this case is the firmware update. The second line indicates where it wants to get the firmware package from. The third line indicates the method it is using to receive the package, which in this case is `wget`. The blue section following shows the manufacturer's server's response to the firmware update fetch

request, and subsequently the firmware image. Because the firmware update was fetched over a plaintext connection, and the SmartCare uses a standard utility to fetch the update, we decided to fetch the update ourselves. After fetching the update using `wget` and performing a file analysis on it, we were able to find that the firmware update was simply a ZIP archive.

Unzipping the archive allowed us to see the SmartCare's main binary along with `bash` scripts for updating the device and one of the SmartCare's main initialization scripts. Based on the initialization script, the device will set itself up, and then run the device's main binary. Knowing this information, the next step in our analysis was to see how the device handles firmware updates, which involves reverse engineering the SmartCare's binary.

E. SmartCare Binary Analysis

Using binary analysis software, we were able to search through the binary and see how it handles updates. The device utilizes the MQTT protocol in order to communicate securely with the manufacturer's server through an encrypted channel. MQTT is a Publisher/Subscriber protocol, where there is a broker which takes in information from publishers, and pushes the information to subscribers. The subscribers subscribe to topics, which are posted by the publishers. In our case, the SmartCare is a subscriber which communicates to the manufacturer's server to fetch the names of firmware updates, the correct hashes for the updates, commands from the user, and the current time. It also acts as a publisher, sending sensor information back to the manufacturer's server.

In terms of actually performing the firmware update, the device will fetch the package using the information gathered over MQTT. Once received, the device will run an MD5 checksum on the package and compare this hash to the hash provided by the manufacturer over MQTT. If both hashes match, the device will go through with the update. If the hashes do not match, the device will reboot, and start the entire process again. The whole verification mechanism is still under investigation for possible security vulnerabilities.

III. Industrial IoT Device Security Analysis

Similar to commercial IoT devices, smart devices are also widely used in industrial applications. These devices, if compromised, may have a more serious impact than compromised commercial IoT devices. To better understand the security protections in place for industrial IoT devices, we selected the Itron Centron smart meter as the other case study. Figure 6 shows the smart meter.

A. High Level Overview

The primary functionality of this device is to measure a customer's energy usage and report the collected information through an RF channel to a nearby meter reader or to a local substation. This information is then used to charge the customer for their energy usage, and may also be used to get statistics on community energy usage.

Fig. 6. Itron Centron CL200 Smart Meter (credit: Itron)

B. Hardware Analysis

Similar to our work on the home automation device, the first step in our analysis was to analyze the hardware platform of the smart meter. Inside of the device we were able to see a heavy-duty plastic cover, which guarded the main hardware platform. When looking at the hardware platform, we identified that it measures line voltage, measures reference voltages, checks the energy flow direction, energy pulse data, and checks the line frequency. Attached to the main hardware platform is a daughterboard, which is used when a company wants to implement functionality on the meter without having to replace the entire device.

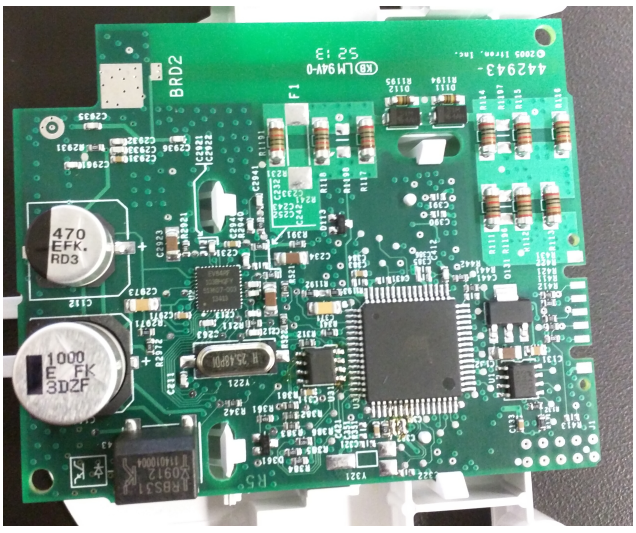

Fig. 7. Smart Meter CL200 Daughterboard

In this case, the daughterboard is used to collect energy

usage information along with tamper data and the ID of the board itself (see Figure 7). Located on the daughterboard is an ATMega microcontroller, a tamper sensor, and a 1 KB EEPROM. Through the microcontroller we were able to re-enable JTAG, and re-enable write access for on-chip memories.

C. Device ID Modification

For our analysis, our objective was to modify the smart meter ID in order for a meter reader to read the incorrect ID for the device. Upon further analysis, the ID was being stored in the external EEPROM. In order to figure out the ID of the meter, we had to read the ID on the meter itself, which is found on the front of the device underneath the grey cover. By analyzing the EEPROM dump, we were able to find where the ID was stored and change the ID to any arbitrary value.

D. Demonstration

Now that we had modified the ID of the meter, we needed to read the ID of the meter remotely to demonstrate that a smart meter reader will pick up the wrong ID from a modified device. Utilizing a software-defined radio (SDR), we were able to run a TCP server on the SDR and connect it to another program which parses wireless information and displays the ID, the tamper bit status, and the energy usage for the meter. Through the experimental platform, we were able to demonstrate that due to the lack of proper protection, one compromised smart meter can "represent" itself as any other smart meter. Figure 8 shows the SDR output in which two smart meters share the same ID but different power consumption values. At the bottom of the figure, there is a meter which identifies as the other, however its power consumption is different than those above it. Through this vector, energy theft becomes possible.

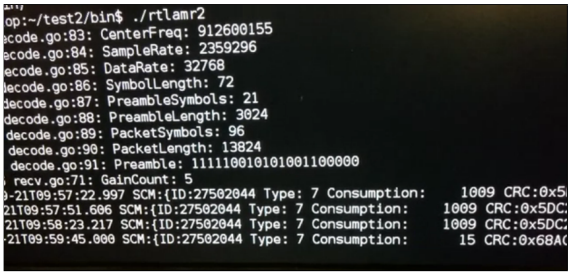

Fig. 8. Demonstration of the security vulnerability on the meter

IV. Discussion

A. Security Impact

As demonstrated, improperly secured IoT devices may easily be compromised during transport and deployment by remote attackers. The effects of these attacks could range from simple backdoors to a total compromise of the device. For example, a compromised Haier SmartCare automation system can lead to property damage or loss, as an attacker can remotely hijack the functionality of the appliances driven by the device. Furthermore, considering the hardware specifications of the SmartCare base unit, an attacker could also inject malicious code into the unit with the objective of discovery and attack of other nodes in the same local network, acting as a beachhead or bridge to other devices.

As it provides a rich operating system environment, a device such as the Haier SmartCare can be used to deploy rogue services in a local network, disrupting the operation of other network nodes. The device could also participate in Address Resolution Protocol (ARP) based attacks, masquerading as the router, allowing the capture of a targeted computer's network traffic.

Industrial devices pose an even bigger threat if compromised, as critical infrastructure may be damaged. In our proof of concept, we were able to change the identification of a smart meter, thus broadcasting energy consumption while masquerading as a different device. In doing this operation, an attacker is capable of altering how billing is performed for their power consumption. Furthermore, as the meter exposes programming and debug interfaces, it can be modified to report reduced reads, further exacerbating the problem. This has a detrimental effect on the economy, as the energy supplier is no longer able to bill customers as it should; the power grid, as expected capacity computations based on recorded values are now incorrect; and the environment, as energy can now be used without proper recording.

B. Safety Concerns

On-field deployment of compromised IoT devices will lead to safety complications. Due to the services these units provide, an attacker can utilize these devices to cause physical harm to users [15]. Compromised industrial IoT devices, such as the Centron CL200 Smart Meter, may be used to harm cyber physical systems, such as the power grid. Excess unmonitored power consumption can result in the overload of the grid, causing outages and in extreme cases, equipment failure.

C. Privacy Concerns

As a home automation system, the Haier SmartCare offers a motion sensor, a magnetic sensor which can be used to determine if a door has been opened and a remote switch to turn devices on and off. A compromised Haier SmartCare can be used by an attacker to build a profile of its user, being able to determine whether the user is at home and possibly some of the user's habits. The leakage of this information causes privacy concerns for users.

D. Device Security Enhancement

Blocking access from a UART console would be a step towards device security. The Haier SmartCare allows un-

restricted access to UART, which was used to change the boot parameters of the unit and extract the login information. Furthermore, utilizing better hashing algorithms for the passwords in the device would increase security if a password login was desired. The current security mechanism in the Haier SmartCare is capable of providing only up to 53 bits of entropy, making the password trivial to brute force. Encrypting the filesystem is another step toward securing the unit, as this would allow direct modifications of the filesystem. An attacker could be able to intercept the device in transport and directly modify the contents of NAND flash, injecting their own payload.

Extra protection must be added to devices that load binaries into a userland. There are two main approaches to this matter. One is to only load and execute signed binaries. This requires the kernel to have a custom loader that verifies binaries before they are executed. Another approach is to encrypt the filesystem. Through this method, an attacker will not be able to externally modify it without first decrypting it.

Further, in devices whose architecture is self contained, that is, microcontroller based systems, it becomes necessary to secure all update channels. External reprogrammability of the microcontroller and any debug interfaces it may feature must be under protection. The microcontroller must also be programmed before being placed in the circuit board, as to avoid adding unnecessary interfaces which could expose functionality. Any external data that is loaded into the microcontroller must also be authenticated. The Itron Centron CL200 smart meter loaded the device descriptors from an external EEPROM, trusting the read values. Furthermore, in the case of the smart meter, the communications channel should also be encrypted for extra security.

V. Conclusion

Through two case studies, we have demonstrated that both commercial and industrial IoT devices are vulnerable to IoT specific attacks. The fact that these devices are of limited protection is a warning that we should take security into consideration when building modern IoT devices. The flooding of IoT devices over the next decade would cause serious security and privacy concerns if we continue utilizing the existing IoT device design flow. Moving forward, we will try to come up with IoT specific design-for-security methodologies to protect IoT devices.

Acknowledgement

This work was supported in part by the Florida Cybersecurity Center (FC2) 2014 Collaborative Seed Grant Program and the Southeastern Center for Electrical Engineering Education (SCEEE 15-001). Mr. Orlando Arias and Mr. Khoa Hoang are partly supported by the REU Supplement of an National Science Foundation grant (CNS-1319105).

References

[1] D. Evans, "The internet of things - how the next evolution of the internet is chaging everything," *White Paper. Cisco Internet Business Solutions Group (IBSG)*, 2011.

[2] P. Middleton, P. Kjeldsen, and J. Tully, "Forecast: The internet of things, worldwide, 2013," *Gartner*, 2013.

[3] D. Welch and S. Lathrop, "Wireless security threat taxonomy," in *Information Assurance Workshop, 2003. IEEE Systems, Man and Cybernetics Society*, 2003, pp. 76–83.

[4] R. Roman, P. Najera, and J. Lopez, "Securing the internet of things," *Computer*, vol. 44, no. 9, pp. 51–58, 2011.

[5] R. Roman, J. Zhou, and J. Lopez, "On the features and challenges of security and privacy in distributed internet of things," *Computer Networks*, vol. 57, no. 10, pp. 2266–2279, 2013.

[6] A. Williams, "How the internet of things helps us understand radiation levels," 2011, [Online]. http://readwrite.com/2011/04/01/ow-the-internet-of-things-help.

[7] D. Viehland and F. Zhao, "The future of personal area networks in a ubiquitous computing world," *International Journal of Advanced Pervasive and Ubiquitous Computing (IJAPUC)*, vol. 2, no. 2, pp. 30–44, 2010.

[8] H. Schaffers, N. Komninos, M. Pallot, B. Trousse, M. Nilsson, and A. Oliveira, "Smart cities and the future internet: Towards cooperation frameworks for open innovation," in *The Future Internet*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2011, vol. 6656, pp. 431–446.

[9] P. N. Mahalle, B. Anggorojati, N. R. Prasad, and R. Prasad, "Identify authentication and capability based access control (IACAC) for the internet of things," *Journal of Cyber Security and Mobility*, vol. 1, pp. 309–348, 2013.

[10] Y. Challal, "Internet of things security: towards a cognitive and systemic approach," Ph.D. dissertation, 2012.

[11] A. Riahi, Y. Challal, E. Natalizio, Z. Chtourou, and A. Bouabdallah, "A systemic approach for IoT security," in *2013 IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS)*, 2013, pp. 351–355.

[12] A. Riahi, E. Natalizio, Y. Challal, N. Mitton, and A. Iera, "A systemic and cognitive approach for IoT security," in *2014 International Conference on Computing, Networking and Communications (ICNC)*, 2014, pp. 183–188.

[13] O. Arias, J. Wurm, K. Hoang, and Y. Jin, "Privacy and security in internet of things and wearable devices," *IEEE Transactions on Multi-Scale Computing Systems*, (to appear).

[14] G. Hernandez, O. Arias, D. Buentello, and Y. Jin, "Smart Nest Thermostat: A smart spy in your home," in *Black Hat USA*, 2014.

[15] D. Halperin, T. Heydt-Benjamin, B. Ransford, S. Clark, B. Defend, W. Morgan, K. Fu, T. Kohno, and W. Maisel, "Pacemakers and implantable cardiac defibrillators: Software radio attacks and zero-power defenses," in *IEEE Symposium on Security and Privacy (SP)*, 2008, pp. 129–142.