An End-to-End View of IoT Security and Privacy

Zhen Ling*, Kaizheng Liu*, Yiling Xu*, Yier Jin†, Xinwen Fu‡
*Southeast University, Email: {zhenling, kzliu, ylxu}@seu.edu.cn
†University of Florida, Email: yier.jin@ece.ufl.edu
‡University of Central Florida, Email: xinwenfu@ucf.edu

***Abstract*—In this paper, we present an end-to-end view of IoT security and privacy and a case study. Our contribution is twofold. First, we present our end-to-end view of an IoT system and this view can guide risk assessment and design of an IoT system. We identify 10 basic IoT functionalities that are related to security and privacy. Based on this view, we systematically present security and privacy requirements in terms of IoT system, software, networking and big data analytics in the cloud. Second, using the end-to-end view of IoT security and privacy, we present a vulnerability analysis of the Edimax IP camera system. We are the first to exploit this system and have identified various attacks that can fully control all the cameras from the manufacturer. Our real-world experiments demonstrate the effectiveness of the discovered attacks and raise the alarms again for the IoT manufacturers.**

I. Introduction

IoT can be defined as interconnecting various uniquely addressable objects through communication protocols. We can interconnect anything including virtual objects together and access those things remotely [1], [2]. IoT has broad applications, including healthcare, life sciences, municipal infrastructure, smart home, retail, manufacturing, agriculture, education and automation. Forbes reported that by 2020 annual revenue of IoT vendors could exceed $470B, Industrial Internet of Things (IIoT) will exceed 60 trillion in the next 15 years and IoT market size was about 900M in 2015 [3]. According to Gartner's hype cycle of emerging technologies in 2016 [4], the expectation for IoT is very high and standardization of IoT platforms will need 5-10 years. The IEEE P2413 Working Group has been trying to standardize the IoT framework since 2014 while there is no consensus yet [5] .

IoT has attracted hackers. There are two kinds of threats: threats against IoT and threats from IoT. 1. *Threats against IoT*: On Oct. 21, 2016, a huge DDoS attack was deployed against Dyn DNS servers and shut down many web services including Twitter [6]. Hackers exploited default passwords and user names of webcams and other IoT devices, and installed the Mirai botnet [7] on compromised IoT devices. The huge botnet was then used to deploy the DDoS attack against Dyn DNS servers. Various other IoT devices have been hacked. IP cameras can be hacked through buffer overflow attacks [8]. Philips Hue lightbulbs were hacked through its ZigBee link protocol [9]. SQL injection attacks were effective against Belkin IoT devices [10]. 2. *Threats from IoT*: Researchers also find cross-site scripting (XSS) attacks that exploited the Belkin WeMo app and access data and resources that the app can access [10].

In this paper, we present an end-to-end view of IoT security and privacy. Our contribution is twofold. First, we present our end-to-end view of an IoT system. We identify 10 basic IoT functionalities related to security and privacy. Based on this view, we systematically analyze the security and privacy requirements in terms of five dimensions: hardware, operating system/firmware, software, networking and big data analytics in the cloud. Second, using the end-to-end view of IoT security and privacy, we present an attack against the Edimax IP camera system. We are the first to exploit this system and have identified various attacks that can fully control all Edimax cameras of the model of interest. The exploit of the camera system demonstrates the usefulness of our view of IoT security and privacy.

The rest of the paper is organized as follows. We introduce our end-to-end view of IoT security and privacy in Section II. In Section III, we introduce the communication protocol of the Edimax camera system. Then we present our exploit of the system in Section IV. We evaluate the exploit in Section V and conclude the paper in Section VI.

II. Framework of IoT Security and Privacy

In this section, we first present our end-to-end view of an IoT system and then present security and privacy requirements for an IoT system.

A. End-to-End View of IoT

We will focus on a standalone IoT system as shown in Figure 1. Such a system normally has three basic components: *thing*, *controller* and *cloud*. The thing is connected to the Internet. For a smart home system, the *thing* is normally behind a wireless router, which adopts NAT to set up a local network of home systems. The *controller* can be a program on a PC or app on a smart device such as a smartphone or tablet. Without loss of generality, we often use a smartphone as an example controller in this paper. Within the local network, the controller can communicate with the thing through the router. However, if the controller is outside, it will not be able to contact the thing directly since the thing is behind NAT (unless port forwarding is enabled on the home router for the thing). Therefore, most IoT systems use a cloud as an intermediate relay between the thing and controller. The thing builds a permanent connection to the cloud. The controller controls or requests information from the thing through the cloud.

We have identified 10 basic functionalities in a IoT system.

978-1-5090-5019-2/17/$31.00 ©2017 IEEE

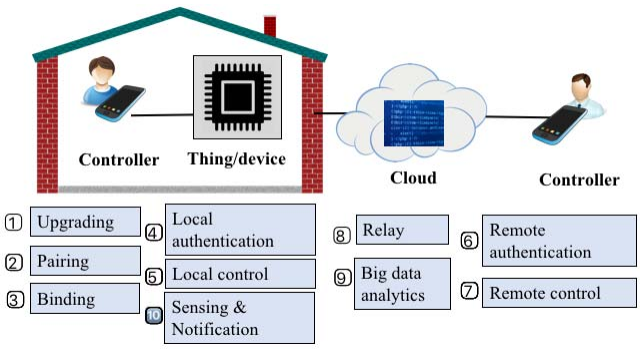

Fig. 1. End-to-end View of IoT

1) **Upgrading**: The firmware of the thing can be upgraded to provide more and better services or a security patch can be applied. The firmware can be a full-fledged embedded Linux system. If the thing is a microcontroller (MCU), the firmware can be a piece of dedicated code for simple control or sensing. For example, a MCU can be used to turn on and off an air conditioner.
2) **Pairing**: Bootstrapping a thing generally involves two steps, pairing and then binding. A controller like a smartphone should be able to communicate with a thing at the bootstrapping time. Such a communication channel can be WIFI, Bluetooth, ZigBee, barcode/scanner, and near field communication. This *connecting process* is denoted as pairing. For example, when many smart things on market are powered on, they behave as a wireless router and allow the controller to connect to the things in order to configure the things. Apparently, if a thing is deployed in public, we have to limit who can access the thing and configure it.
3) **Binding**: This is the process of configuring the thing through the controller once pairing is done. The controller may bind the thing to the Internet, that is, connect the thing to the Internet. For example, the controller can require a user to input the WiFi SSID (Service Set Identifier) and password of a wireless router and send the information to the thing, which can then connect to the Internet. Another important binding activity is to bind the thing and its users. For example, the controller can learn the identity of the thing (e.g., the MAC address of the wireless interface on the thing) via the communication channel used in the pairing process. Therefore, the user and the thing can be bound together via an appropriate protocol.
4) **Local authentication**: Within a local network, the controller may connect to a port open on a thing, which should authenticate the user and then allow further actions from the user.
5) **Local control**: Once a user is authenticated, the controller can send commands to control the thing.
6) **Remote authentication**. If the controller is on the Internet and not in the local network, it may not be able to directly contact the thing, which may be behind NAT, and has to go through the cloud for authentication.
7) **Remote control**: If the controller is on the Internet and not in the local network, it may have to control the thing through a cloud.
8) **Relay** by cloud: For remote authentication and control, the cloud is to relay the authentication and control messages between the thing and controller. The cloud may have an authentication server to authenticate both the thing and controller and connect them together.
9) **Big data analytics** by cloud: The cloud may collect the data from things and users, and perform big data analytics. A cloud may connect to other clouds that serve other things, share data and request further analytics capabilities.
10) **Sensing and notification**: Many things are smart. For example, a thing may sense the room temperature and notify the user if the temperature is too low or high. A thing can also notify the user about abnormal behaviors such as too many login attempts on the thing.

B. Security and Privacy in IoT

To secure an IoT system, we have to consider five dimensions: hardware, operating system/firmware, software, networking and data generated and maintained within the system, as shown in Figure 2. As illustrated in Figure 1, an IoT system has quite a few components, all of which should be inspected from these five aspects. The 10 functionalities of IoT identified in Section II-A span across these five dimensions. We have to secure any interface that may interact with users (including attackers) in an IoT system.

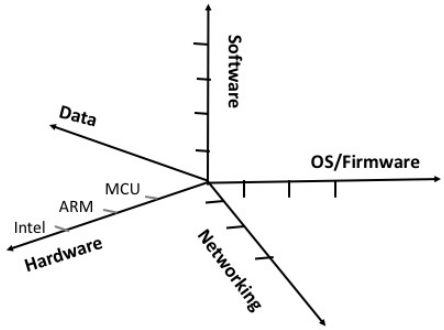

Fig. 2. Five Aspects of IoT Security and Privacy

Hardware security: Hardware security is critical when attackers can physically access the IoT devices. For example, many IoT devices do not disable their debugging ports after the testing and validation stage, which give attackers full access to the internal firmware. In fact, almost all IoT devices have hardware vulnerabilities which may be exploited by attackers including the UART/JTAG debugging ports, multiple boot options, and unencrypted flash memory [11]–[13]. Through the hardware backdoors, attackers can easily bypass software level integrity checking by either disabling the checking functionality or booting the system through an injected firmware image. An IoT security vulnerability database is recently constructed, which presents a large spectrum of different types of vulnerabilities including hardware security related vulnerabilities

[14]. Accordingly, countermeasures are recently proposed to prevent physical attacks such as runtime attestation to prevent TOCTOU attacks [15]. TPM [16], TrustZone [17] and Intel SGX [18] can provide hardware-level security.

Operating system (OS)/firmware and software security and privacy: Given the often limited functionalities of an IoT device, a trustworthy operating system [19] can be implemented on the device if the cost is permitted. The control app for a thing is often installed on a smartphone and software secure measures should be applied in order to prevent the attack against the app like the attack in [10]. We can also not blindly trust the cloud for security. For example, servers installed on Amazon EC2 have to be secured by whoever deploys the servers. Software security issues are similar to those in the traditional computer systems. For example, backdoors and public and private SSL key pairs are discovered by performing static analysis on a large number of unpacked firmwares [20]. Chen *et al.* [21] perform large-scale automated dynamic analysis of various firmwares to discover potential exploits using the Metasploit Framework. A case study of a firmware modification attack is investigated in [22]. A buffer overflow exploit is found by analyzing Home Network Administration Protocol (HNAP) [23] so that it can be used to execute any code on the device. A stack-based buffer overflow of the general library, glibc [24], is exploited to attack several home hubs [25].

Network Security and Privacy: An IoT system is a networked system and the whole system has to be secured from end to end [26]–[32]. Communication should be encrypted to prevent the leak of sensitive information. Authentication has to be carefully implemented. We have differentiated pairing from binding. Recall that in the pairing process, the controller needs to connect to the IoT device in order to configure the thing. However, most IoT devices allow any controller in proximity for pairing. The risk of such practice may be small in a private setting like a home. However, for a large-scale deployment in a public environment, anybody with access to the devices can reconfigure the system and may break into the system. After pairing, we run the binding process to bind identities to the thing in order to control it. The authentication has to be set up in a proper way. For example, weak passwords should be avoided. An IoT system may be composed of a large number of nodes with sensing capabilities and security techniques in sensor networks can be applied accordingly [33]–[36].

Many manufacturers fail to provide necessary protection for their networked IoT devices, which are under constant attacks nowadays. The Mirai DDoS attack [7] was possible because of the weak passwords on various IoT devices. Rouf *et al.* [26] exploit the unsecured wireless communication protocol of automatic meter reading. Dhanjani [27] hacks the Phillips Hue lightbulb system and finds that the authentication mechanisms are not strong. Molina [28] exploits the KNX, a standardized home automation communication protocol, and finds that the lack of authentication and encryption allows an attacker to remotely control the appliances in a hotel. Rahman *et al.* [29] find the communication protocol vulnerabilities

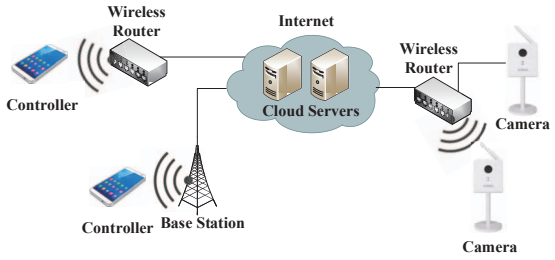

Fig. 3. Architecture of the camera system

of the wearable device (Fitbit). By automatically analyzing the applications and forging the authentication messages, Zuo *et al.* [32] design an authentication message generator to perform brute force attacks against the corresponding remote application server. Obermaier and Hutle [31] investigate the vulnerabilities of communication protocols of four surveillance camera systems.

Big Data Analytics: Since the cloud sits between the controller and IoT devices, it can collect all the data. Many of the systems including Amazon AWS IoT are set up in this way. We have to question: should the cloud know everything and collect data about us and our belongings? For example, for remote authentication, should the cloud serve as the authentication server to authenticate controllers/things? However, big data collected by the cloud can help defeat attacks. For example, a proper intrusion detection system over the cloud can prevent another round of Mirai attack. Since things are often very specific, intrusion detection can be made easy.

III. Protocols of Edmiax Cameras

In this section, we present a case study of exploiting an IP camera system manufactured by Edimax under our view of IoT security and privacy. We first introduce the architecture of the camera system and then present the detailed communication protocol.

A. Architecture of the Camera System

By traffic analysis, we find that the Edimax camera system has three components, including the camera, controller, and cloud servers as shown in Figure 3. If the controller and camera are in the same local network, the controller can communicate with the camera locally and fetch the live video through a web server on the camera. In this paper, we concentrate on remote attacks and will focus on remote communication protocols of the camera system when the controller and camera are not in the same local network. The camera connects to the Internet through an ethernet cable or WiFi. The controller can be an app on a smartphone. The controller communicates with the camera via the cloud servers, including *the registration server* and *the command relay server*. The registration server is used for device registration for both the controller and the camera. The command relay server forwards command messages between them.

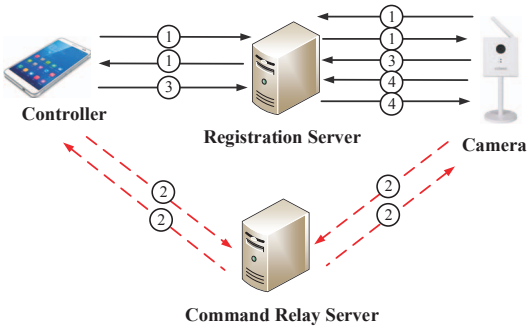

Fig. 4. Registration phase

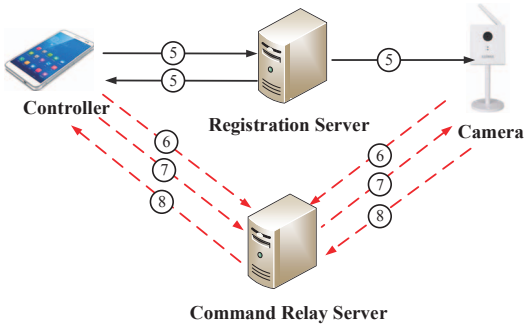

Fig. 5. Connection establishment phase and data communication phase

B. Paring, Binding and Registration

We first investigate the paring process. When the camera is used for the first time, a user needs to connect it to her home network using an ethernet cable. The software *EdiView Finder Utility* should be installed on a computer in the same home network. This utility is used to search the home wireless router and configure the camera to use the home wireless router. At this point, the wired connection of the camera can be disconnected.

In the binding process, we can change the password and other configurations such as the resolution of the image via a web page of the camera. The link is *http://host/setup.asp?r=20141126*, where *host* is the local IP address of the camera. Upon connecting to the Internet, the camera registers with two remote servers, i.e., registration server and command relay server. The controller also registers with both servers.

The packets transmitted between the controller, camera, and remote servers are obfuscated instead of encrypted. The right shift is performed over all characters but the first character in packets. The number of positions in the right shift is between 1 and 7, which is the difference between the first original character and the corresponding obfuscated character. The first original character is always "<", since a pair of "<>" is used to delimit key and value pairs. When the camera or controller receives a packet, it compares the first character with "<" to obtain the number of positions and perform the corresponding left shit to obtain the plaintext.

At the registration phase, all the packets use UDP. The UDP service ports of both the registration server and the command relay server are 8760. Figure 4 illustrates the registration procedure for both the camera and controller. Since the first three steps of both the camera and controller are similar, we take the camera as an example to present the detailed procedure as follows.

STEP 1: In this step, the camera registers with the registration server. The camera first sends a UDP packet to the registration server. The packet has a value of "1" in the "opcode value" field, referred to as *command value* in this paper, and a UUID (Universally Unique Identifier) in the "id value" field. The UUID is used to uniquely identify the connection.

Upon receiving the packet with the command value "1" from the camera, the registration server responds with a UDP packet with a command value "10". The response packet consists of the UUID received from the camera, and the IP addresses and the ports of both the camera and the command relay server. Consequently, the camera can learn the IP address and port of command relay server from this response packet.

STEP 2: In this step, the camera registers with the command relay server. The camera first sends a UDP packet to the command relay server with a command value "1" and a new UUID to uniquely identify this connection. The command relay server responds with a UDP packet with a command value "10". The packet contains the UUID received from the camera, and the IP addresses and the ports of both the camera and the command relay server.

STEP 3: Once the camera receives response from the command relay server, it sends a packet with a command value "2" back to the registration server. This packet contains a new UUID. It is used to inform the registration server the fact that it has registered with the command relay server.

STEP 4: The camera sends two successive UDP packets to the UDP service port 8765 of the registration server. The first packet with a code value of "3000" is used to inform the registration server that the camera is online. The second packet with a code value of "1010" carries the camera information such as the camera model, MAC address, type, alias, LAN IP address and port of this camera, serial number, camera firmware version, and camera status.

After receiving the messages with the code value of "1010" from the camera, the registration server responds with a UDP packet with a code value of "1020". The packet contains the MAC address and the status of the camera. The camera repeats **STEP 1** to **STEP 4** around every 20 minutes to inform the registration server that the camera is online.

C. Camera Discovery Phase and Authentication

In the camera discovery phase, the controller tries to first check the online status of the camera via the registration server as shown in Figure 5 and then sends the authentication information to it. The UDP service ports of the registration server for the camera and the controller are 8765 and 8766, respectively.

STEP 5: In this step, a user sets the configuration of the controller in order to check the online state of a specified

camera. The user inputs an alias of the camera, the MAC address of the specified camera, and the password through the graphic user interface of the controller. The controller then sends two successive UDP packets to the registration server. The first packet with a code value of "3000" is to inform the registration server that the controller wants to check the state of the camera. The second packet with a code value of "2030" contains the MAC address of the camera and the information of the controller, including the LAN IP address and port, the device firmware version, and a relay ID generated by the controller. The relay ID is composed of the camera's MAC address and a timestamp. It is used in the data communication phase to correctly interconnect the two TCP connections from a pair of controller and camera on the command relay server.

After receiving the request from the controller, the registration server checks the camera status first. If the camera is offline, the registration server responds with a packet with a code value of "5000". Otherwise, the registration server responds with a packet with a code value of "2040" to the controller. This packet includes the IP addresses and ports of both the camera and the command relay server, the relay ID, camera firmware version, model, type, alias, and camera status. The registration server also adds extra messages to the "2030" packet and changes the code value to "2020", and then forwards it to the camera. The extra messages of "2020" packet include the IP addresses and ports of the camera, the controller, and the command relay server. Therefore, the camera can learn the relay ID from this packet.

D. Remote Data Communication Phase

There are two ways for the controller to control the camera remotely. First, the controller and the camera try to directly communicate with each other using the UDP protocol. Second, If the attempt of a direct UDP connection fails, the controller and the camera will communicate with each other via the command relay server using the TCP protocol. In this paper, we mainly concentrate on the data communication using TCP.

STEP 6: To communicate with TCP, both the camera and the controller establish TCP connections to the command relay server. Recall that the camera and the controller obtain the IP address and ports of the command relay server from the registration server. The camera also obtains the relay ID generated by the controller through the registration server. Both the camera and the controller send a TCP packet that contain the MAC address of the camera and the relay ID to the command relay server. According to the MAC address of camera and the relay ID, the command relay server can interconnect these two TCP connections and relay the data between the camera and the controller. However, the command relay server does not send any response packets to neither the camera nor the controller.

STEP 7: To obtain live images from the camera, the controller sends requests to the camera via the command relay server. The request packets contain a value of "/mobile.jpg" in "url value" field, and authentication information in "auth value" field. The authentication information in the format of *username*:*password* is encoded in the Base64 scheme. The default username and password are *admin* and *1234*, respectively. Users can change the password through the web page of the camera. However, they cannot change the username as it is hardcoded in the camera. Once the command relay server receives the request packets from controller, it forwards them to the camera.

STEP 8: After the camera receives the request, the camera first checks the authentication information. If the authentication information is correct, the camera sends images back to the command relay server, which forwards them to the controller. Otherwise, the camera will send an authorization failure packet to the controller.

Every time the controller tries to obtain an image, it needs to send the request packet that contains the authentication information. Therefore, the controller repeats the **STEP 7** and **STEP 8** so as to continuously derive the live images taken by the camera.

IV. Security Vulnerabilities of Edimax Cameras

In this section, we first present three remote attacks against the Edimax IP camera of interest: device scanning attack, brute force attack, and device spoofing attack. Using these attacks, we can remotely control any camera.

A. Device Scanning Attack

The attacker can find out all online cameras by enumerating all the possible MAC addresses. Recall the procedure of the connection establishment phase. After the controller sends a "2030" packet, the controller receives a "2040" packet if the camera is online. If the camera is offline, the controller will receive a packet with a code value of "5000". Therefore, the attacker can construct a "2030" packet with the specified camera MAC address, and check whether the specified camera is online according to the response packet.

The MAC address space of a manufacturer can be known from the Internet. A MAC address contains 12 characters. The first 6 characters indicate manufacturer and the other 6 characters indicate the namespace given to the manufacturer. Products of the same model from a manufacturer are usually assigned consecutive MAC addresses. Thus the attacker can infer MAC addresses based on the MAC address of his own purchased camera, enumerate the 12 characters of the MAC address and can verify the state of the camera with each potential MAC address.

B. Brute Force Attack

If a user changes the default password of a camera, the attacker can find the password via a brute force attack. In the data communication phase, when the controller sends a TCP request that contains the authentication information, the camera responds with images if the authentication information is correct. Therefore, the attacker can enumerate all possible passwords by repeating the TCP request, and determine if the password is right or not in terms of the response packet. Our experiments show that the command relay server does

not block this brute force attack. If a user chooses a 4-digit password like the default one, the brute force attack works.

Although there is no explicit password policy from the manufacturer, we find that the camera password can be 63 characters long, and allows digits, special characters, upper-case, and lower-case alphabetic letters. Therefore, if the user employs a long and complicate password, the brute force attack may not work.

C. Device Spoofing Attack

The device spoofing attack can obtain a camera password of any length and combination. In the device spoofing attack, the attacker creates a software bot implementing the camera communication protocol in order to emulate the camera. When the user opens the control app, the TCP request packet with the password is sent to the attacker's software bot. Therefore, the attacker obtains the password.

The detailed attack process is presented as follows.

1) The attacker chooses an online camera that uses a non-default password based on the device scanning results and creates the software bot with the specific MAC address. Any camera from this manufacturer can be spoofed this way.
2) The software bot registers with the registration server and the command relay server by performing **STEP 1** to **STEP 4**. The software bot sends two UDP packets with the command value of "1" and "2" to both registration server and command relay server for registration. It then sends two successive UDP packets (i.e., code value of "3000" and code value of "1010") to the registration server informing the server that the spoofed camera is online. Once the software bot receives the packet with the code value of "1020" from the registration server, the attacker knows that the spoofed camera is online. The software bot repeats **STEP 1** to **STEP 4** as many times as possible, since the real camera will register itself by performing **STEP 1** to **STEP 4**.
3) When the user opens the control app, the app sends two successive UDP packets (i.e., code value of "3000" and code value of "2030") to the registration server as introduced in **STEP 5**. The registration server forwards the packets to the software bot spoofing the camera. Simultaneously, the registration server informs the control app that the camera is online.
4) The control app builds a TCP connection to the command relay server and sends a TCP request to the server automatically. The command relay server forwards the TCP request that contains the authentication information to the software bot. Recall that the authentication information is encoded with the Base64 scheme and the format is *username*:*password*. As a result, it is trivial for the attacker to derive the password from the authentication information.
5) The spoofed camera should be offline as soon as it obtains the authentication information. Recall that the real camera registers with the registration server and the command relay server every 20 minutes. Accordingly, it takes at most 20 minutes for the real camera to get online again after the spoofed camera obtains the authentication information. After that, the user can see the images and videos taken by real camera again and may not realize that the camera has been compromised. Once the attacker obtains the password, she can fully control the camera.

V. Evaluation

In this section, we present our experiment results validating the three attacks against Edimax IP cameras. All the attacks were performed over our own purchased cameras.

To verify the feasibility of the device scanning attack, we first put our Edimax IP camera online. We then send a packet with a code value of "2030" to the Edimax registration server and receive a packet with a code value of "2040". We then put the camera offline. We resend a packet with a code value of "2030" to the registration server and receive a packet with a code value of "5000". Therefore, we can scan any potential MAC address to determine if the corresponding camera is online or not.

To verify the brute force attack, we set a random 4-digit password for our own camera. We then run the brute force attack and can identify the right password in a few minutes.

We now present the results of evaluating the device spoofing attack. The device spoofing attack may fail if the real camera registers with the registration server and this kicks our spoofed camera offline. In such a scenario, the controller will connect to the real camera, and the spoofed camera cannot receive the request packet with the authentication information from the controller. However, our software bot spoofing the camera can send out the registration packet continuously in order to increase the attack success rate. To verify our attack, we connect the real camera to the Internet, and the spoofed camera registers with the registration server every 10 seconds. A user opens the controller randomly during the attack. If the spoofed camera receives the authentication information, the attack succeeds; otherwise, it fails. We perform the experiments for 50 times and the spoofed camera receives the authentication information 49 times.The success rate of the device spoofing attack is up to 98%.

VI. Conclusion

In this paper, we first present our view of an IoT system that includes the thing, cloud and controller from an end-to-end perspective. 10 basic functionalities have been identified for such a system. Those functionalities have to be secured properly according to our risk analysis of different components of the IoT system. We then present our exploit of an IP camera system and discovered three attacks including device scanning attack, brute force attack and device spoofing attack that can fully control all of the IP cameras from the manufacturer. We performed real-world experiments to validate the attacks and find that the device spoofing attack can obtain a user's password at a probability of 98% whatever the password is.

Our end-to-end view of IoT Security and privacy can serve as the guide to design a secure and privacy preserving IoT system.

ACKNOWLEDGMENTS

This work was supported in part by National Natural Science Foundation of China under grants 61502100, 61532013, 61402104, 61572130, 61602111, 61632008, and 61320106007, by US NSF grants 1642124, 1461060, 1547428, 1723587, and Cisco, by Jiangsu Provincial Natural Science Foundation of China under grants BK20150637 and BK20 140648, by Jiangsu Provincial Key Technology R&D Program under grants BE2014603, by Jiangsu Provincial Key Laboratory of Network and Information Security under grants BM2003201, by Key Laboratory of Computer Network and Information Integration of Ministry of Education of China under grants 93K-9 and by Collaborative Innovation Center of Novel Software Technology and Industrialization. Any opinions, findings, conclusions, and recommendations in this paper are those of the authors and do not necessarily reflect the views of the funding agencies.

REFERENCES

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, Oct. 2010.

[2] J. Gubbia, R. Buyyab, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, 9 2013.

[3] L. Columbus, "Roundup of internet of things forecasts and market estimates," https://www.forbes.com/sites/louiscolumbus/2016/11/27/roundup-of-internet-of-things-forecasts-and-market-estimates-2016/#27232e3d292d, 2016.

[4] Gartner, Inc., "Gartner's 2016 hype cycle for emerging technologies identifies three key trends that organizations must track to gain competitive advantage," http://www.gartner.com/newsroom/id/3412017, August 2016.

[5] The IEEE P2413 Working Group, "Standard for an architectural framework for the internet of things (iot)," https://standards.ieee.org/develop/project/2413.html, 2016.

[6] S. Hilton, "Dyn analysis summary of friday october 21 attack," http://dyn.com/blog/dyn-analysis-summary-of-friday-october-21-attack/, October 2016.

[7] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, D. Menscher, C. Seaman, N. Sullivan, K. Thomas, and Y. Zhou, "Understanding the mirai botnet," in *Proceedings of the 26th USENIX Security Symposium (Security)*, 2017.

[8] R. Chirgwin, "Get pwned: Web cctv cams can be hijacked by single http request - server buffer overflow equals remote control," http://www.theregister.co.uk/2016/11/30/iot_cameras_compromised_by_long_url, November 2016.

[9] G. Rabinowitz, "Israeli hackers show light bulbs can take down the internet," http://www.timesofisrael.com/israeli-hackers-show-light-bulbs-can-take-down-the-internet/, November 2016.

[10] E. Kovacs, "Belkin wemo devices expose smartphones to attacks," http://www.securityweek.com/belkin-wemo-devices-expose-smartphones-attacks, November 2016.

[11] 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*, vol. 1, no. 2, pp. 99–109, 2015.

[12] J. Wurm, O. Arias, K. Hoang, A.-R. Sadeghi, and Y. Jin, "Security analysis on consumer and industrial iot devices," in *Proceedings of the 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*, 2016.

[13] Y. Jin, G. Hernandez, and D. Buentello, "Smart nest thermostat: A smart spy in your home," in *Proceedings of the Black Hat USA*, 2014.

[14] S. in Silicon Lab, "Iot security vulnerability database," http://www.hardwaresecurity.org/iot/database, August 2017.

[15] S. Zeitouni, G. Dessouky, O. Arias, D. Sullivan, A. Ibrahim, Y. Jin, and A.-R. Sadeghi, "Atrium: Runtime attestation resilient under memory attacks," in *Proceedings of the International Conference On Computer Aided Design (ICCAD)*, 2017.

[16] A. Tomlinson, *Introduction to the TPM, Smart Cards, Tokens, Security and Applications*. Springer, 2008.

[17] "Arm security technology, building a secure system using trustzone technology," 2009.

[18] "Intel software guard extensions (intel sgx)," https://software.intel.com/en-us/sgx, 2017.

[19] B. Parno, J. M. McCune, and A. Perrig, *Bootstrapping Trust in Modern Computers*, 1st ed. Springer Publishing Company, Incorporated, 2011.

[20] A. Costin, J. Zaddach, A. Francillon, and D. Balzarotti, "A large-scale analysis of the security of embedded firmwares," in *Proceedings of the 23rd USENIX Security Symposium (Security)*, 2014.

[21] D. D. Chen, M. Egele, M. Woo, and D. Brumley, "Towards automated dynamic analysis for linux-based embedded firmware," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2016.

[22] A. Cui, M. Costello, and S. J. Stolfo, "When firmware modifications attack: A case study of embedded exploitation," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2013.

[23] /DEV/TTYS0, "Hacking the d-link dsp-w215 smart plug," http://www.devttys0.com/2014/05/hacking-the-d-link-dsp-w215-smart-plug/, 2014.

[24] "Critical security flaw: glibc stack-based buffer overflow in getaddrinfo() (cve-2015-7547)," https://access.redhat.com/articles/2161461, 2015.

[25] M. Smith, "Security holes in the 3 most popular smart home hubs and honeywell tuxedo touch," http://www.networkworld.com/article/2952718/microsoftsubnet/security-holes-in-the-3-most-popular-smart-home-hubsand-honeywell-tuxedo-touch.html, 2015.

[26] I. Rouf, H. Mustafa, M. Xu, W. Xu, R. Miller, and M. Gruteser, "Neighborhood watch: Security and privacy analysis of automatic meter reading systems," in *Proceedings of the 19th ACM Conference on Computer and Communications Security (CCS)*, 2012.

[27] N. Dhanjani, "Security evaluation of the philips hue personal wireless lighting system," http://www.dhanjani.com/docs/Hacking%20Lighbulbs%20Hue%20Dhanjani%202013.pdf, 2013.

[28] J. Molina, "Learn how to control every room at a luxury hotel remotely," https://www.defcon.org/images/defcon-22/dc-22-presentations/Molina/DEFCON-22-Jesus-Molina-Learn-how-to-control-every-room-WP.pdf, 2014.

[29] M. Rahman, B. Carbunar, and M. Banik, "Fit and vulnerable: Attacks and defenses for a health monitoring device," in *Proceedings of the 34th IEEE Symposium on Security and Privacy (S&P)*, 2013.

[30] Mario Ballano Barcena and Candid Wueest, "Insecurity in the internet of things," https://www.symantec.com/content/dam/symantec/docs/white-papers/insecurity-in-the-internet-of-things-en.pdf, 2015.

[31] J. Obermaier and M. Hutle, "Analyzing the security and privacy of cloud-based video surveillance systems," in *Proceedings of the 2nd ACM International Workshop on IoT Privacy, Trust, and Security (IoTPTS)*, 2016.

[32] C. Zuo, W. Wang, R. Wang, and Z. Lin, "Automatic forgery of cryptographically consistent messages to identify security vulnerabilities in mobile services," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2016.

[33] X. Yao, X. Han, X. Du, and X. Zhou, "A lightweight multicast authentication mechanism for small scale iot applications," *IEEE Sensors Journal*, vol. 13, no. 10, pp. 3693–3701, Oct. 2013.

[34] Y. Xiao, V. Rayi, B. Sun, X. Du, F. Hu, and M. Galloway, "A survey of key management schemes in wireless sensor networks," *Journal of Computer Communications*, vol. 30, no. 11, pp. 2314–2341, Sep. 2007.

[35] X. Du, Y. Xiao, M. Guizani, and H. H. Chen, "An effective key management scheme for heterogeneous sensor networks," *Ad Hoc Networks, Elsevier*, vol. 5, no. 1, pp. 24–34, Jan. 2007.

[36] X. Du and H. H. Chen, "Security in wireless sensor networks," *IEEE Wireless Communications Magazine*, vol. 15, no. 4, pp. 60–66, Aug. 2008.