Security Vulnerabilities of Internet of Things: A Case Study of the Smart Plug System

Zhen Ling, Junzhou Luo, Yiling Xu, Chao Gao, Kui Wu, *Senior Member, IEEE*, and Xinwen Fu

[redacted]

***Index Terms*—Attacks, countermeasures, Internet of Things (IoT), vulnerabilities.**

I. Introduction

THE EMERGENCE of Internet of Things (IoT) provides the capabilities of connecting smart devices, small actuators, and people anywhere and anytime to the Internet. Gartner forecasts that the number of IoT grows 31% from 6.4 billion in 2016 to 8.4 billion in 2017, and will reach 20.4 billion by 2020 [1]. Smart plugs, as one type of fast emerging IoT devices, are gaining increasing popularity in home automation, with which users can remotely monitor and control their homes. Fig. 1 shows an example smart plug, i.e., Edimax SP-2101W, and the iPad that is installed with the control application, i.e., EdiPlug. Various applications can be implemented over such a system. For instance, in winter time users can turn on the heater with a smartphone in advance to warm up their homes before they return home. They can also rely on smart plugs with the energy management function to accurately monitor the energy consumption. Medical equipment may also be connected to smart plugs for smart health. Due to the tremendous benefit, smart plugs have been deployed worldwide in millions of homes.

Security concerns come along with the popularity of smart plugs. Compromised smart plugs would lead to both security and privacy breach of home users. If smart plugs are used in commercial or industrial buildings for demand response [2], the consequences of smart plugs being compromised and controlled by attackers could be disastrous. A disrupted medical equipment connected to smart plugs may threaten a patient's life. In recent years, the security concerns of smart plugs have received substantial consideration in both industry and academic communities [3], [4].

Despite the importance and broad concern of security problems in smart plugs, we found their vulnerabilities are still prominently exposed. As an evidence, we in this paper case study the security problems of a typical smart plug system, i.e., Edimax SP-2101W. With reverse engineering, we disclose its entire communication protocols and identify its vulnerabilities that could open the door to different attacks. We propose four attacks: 1) device scanning attack; 2) brute force attack; 3) spoofing attack; and 4) firmware attack. Extensive real-world experiments show that we can effectively and efficiently obtain a victim's authentication credentials via these attacks. As a remedy, we present defense guidelines to mitigate these attacks.

The goal of this paper is to send out a strong message to the IoT community and hopefully to enforce smart plug manufacturers/developers to put security at a higher priority. As such, the code of our attacks will not be disclosed.

Our main findings regarding the vulnerabilities of the Edimax plug system in question can be summarized as follows.

A. Insecure Communication Protocols

Sswse the communication protocols do not rely on cryptographic mechanisms, an attacker could capture network traffic and reverse engineer the communication protocols. In this case, the system is subject to various eavesdropping attacks.

Manuscript received January 14, 2017; revised May 3, 2017; accepted May 7, 2017. Date of publication May 23, 2017; date of current version December 11, 2017. This work was supported in part by the National Natural Science Foundation of China under Grant 61502100, Grant 61632008, Grant 61402104, Grant 61572130, Grant 61602111, Grant 61532013, and Grant 61320106007, in part by the National Science Foundation under Grant 1461060, Grant 1642124, and Grant 1547428, in part by the Natural Sciences and Engineering Research Council of Canada under Grant 261409-2013, in part by the Jiangsu Provincial Natural Science Foundation of China under Grant BK20150637 and Grant BK20140648, in part by the Jiangsu Provincial Key Laboratory of Network and Information Security under Grant BM2003201, in part by the Key Laboratory of Computer Network and Information Integration of Ministry of Education of China under Grant 93K-9, and in part by the Collaborative Innovation Center of Novel Software Technology and Industrialization. *(Corresponding author: Zhen Ling.)*

Z. Ling, J. Luo, and Y. Xu are with the School of Computer Science and Engineering, Southeast University, Nanjing 211189, China (e-mail: zhenling@seu.edu.cn; jluo@seu.edu.cn; ylxu@seu.edu.cn).

C. Gao and X. Fu are with the Department of Computer Science, University of Massachusetts at Lowell, Lowell, MA 01854 USA (e-mail: cgao@cs.uml.edu; xinwenfu@cs.uml.edu).

K. Wu is with the Department of Computer Science, University of Victoria, Victoria, BC V8P 5C2, Canada (e-mail: wkui@uvic.ca).

Digital Object Identifier 10.1109/JIOT.2017.2707465

B. Lack of Device Authentication

The remote server used by an app communicating with plugs does not authenticate the plugs. This widely opens the door for an attacker to perform our four attacks.

1) The Edimax plug system uses the MAC address of a plug as the identity of the plug. We are able to use the *device scanning attack* and scan the MAC address space of the vendor in order to find the online status of all smart plugs made by the vendor. The device scanning attack can also reveal if users use the default password of a plug since many users do not change the default password of their smart devices [3] due to lack of security awareness.
2) If the plug is online and the password is changed, we can perform the *brute force attack* to infer the passwords. Given a default password of "1234," it is likely that a user may change it to a four-digit one since the vendor does not explicitly list their password policy in their documentation. The remote server does not limit the password attempts by an app.
3) If long passwords are employed by users, we can launch the *device spoofing attack*, which blocks the genuine plug and pretends to be a legal one, waiting for the remote application to send the authentication credential of a user for login and use of the plug. In this attack, the users leak their authentication credentials once opening the plug control applications. The attack is also stealthy and the users can hardly realize that they are attacked. Using the credential, the attacker can completely control the genuine plug.
4) Moreover, we study the firmware update process and perform the *firmware attack* to upload a malicious firmware to the plug. With such a malicious firmware, an attacker can create a reverse tunnel from the plug to a desired server and gain the root access on the plug system.

For countermeasures to the potential attacks exploiting the above vulnerabilities, we present the following guidelines to protect smart plug systems, including secure communication protocols to block eavesdropping attacks, mutual authentication between the control app and plug through the remote server, intrusion detection system for abnormal behavior detection, anti-bot mechanisms, and validation of data integrity.

The rest of this paper is organized as follows. We give an overview of our protocol analysis strategy and the discovered Edimax plug system architecture in Section II. In Section III, we present the detailed communication protocol, including the registration phase, authentication phase, control phase, and firmware update. In Section IV, we introduce four attacks. In Section V, we perform extensive empirical experiments to demonstrate the feasibility and effectiveness of our attacks. In Section VI, we discuss the corresponding countermeasures. Related work is presented in Section VII. Finally, we conclude this paper in Section VIII.

II. Protocol Analysis Overview

In this section, we first briefly introduce the smart plugs we will exploit. We then introduce our platform that is used to analyze this Edimax plug system. We also discuss strategies to analyze the content of the communication traffic. Finally, we introduce the big picture of the smart home plug system architecture of interest.

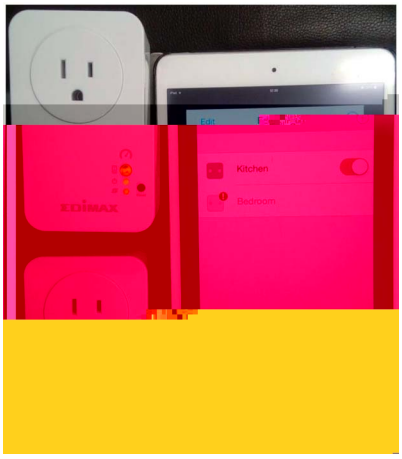

Fig. 1. Edimax plug and iPad installed with the control application.

A. Smart Plugs of Interest

A smart home plug is an electric device that can be plugged into an ordinary outlet. It provides outlets for other electronic devices, e.g., lamps and fans. It is often designed to connect to the wireless home network so that a user can install an app on her smart device, e.g., smartphone, and control the electronic device plugged into the smart plug over the Internet. Smart plugs are gaining popularity for building a home automation system.

We selected a typical smart plug, i.e., Edimax SP-2101W, as shown in Fig. 1. The device is available from Amazon and Walmart and has a rating of more than 4.0 out of 5.0. The app for controlling the plug supports both Android and iOS platforms. The plug provides the power meter functionality and allows users to manage the energy consumption. For instance, a user can monitor the power usage of the plugged appliance and make a schedule to turn on/off the appliance via the app. Moreover, a user can set up email information, including username, password, simple mail transfer protocol (SMTP) server, etc., for the plug so that the plug can send alert emails to the users.

B. Network Traffic Acquisition and Analysis Platform

To analyze the network traffic and learn the architecture of the Edimax plug system, we establish an experiment network platform to capture the traffic at both smart plug and app. We use two machines to set up two wireless APs with wireless USB adapters. We install the Ubuntu 14.04 operation system on these two machines, which are connected to the Internet through Ethernet network cards to obtain public IP addresses. To establish wireless local area network (WLAN), the network address translation (NAT) function is configured using the Linux firewall, i.e., iptables. Moreover, we set up a dynamic host configuration protocol (DHCP) service to automatically assign local IP addresses to the devices connected to

Step 1: The smart plug establishes a TCP connection to www.google.com. In this step, the smart plug performs the network reachability test to check if the plug can access the Internet. If it could not access the Google website, the plug will stop working. Apparently, this strategy of testing Internet connection is not robust since a number of countries such as China block Google services [5].

Step 2: The smart plug connects to a time server pool.ntp.org, using the network time protocol (NTP) to synchronize the clock of the plug. A synchronized clock is necessary since the plug system uses the time information for its communication and task scheduling service, e.g., periodically turning on/off the plug on time.

Step 3: The smart plug sends datagram packets to a remote server and registers with the server. According to our analysis, this remote server is deployed on Amazon EC2 and is only used for relaying UDP packets for authentication, we call it the *authentication server*. The UDP port of the authentication server for the plug is 8765.

The smart plug sends two consecutive UDP packets to the authentication server. The content in these packets are encapsulated using the XML format. The first datagram packet sent by the plug contains a value of "3000" in the "code value" field to inform the authentication server for the registration. This field is referred to as "command type" in this paper. The second datagram packet includes a command type of "1010" and the plug information including the plug model, *MAC address*, type, alias, LAN IP address and port of this plug and device firmware version. The second packet is used to notify the authentication server that the plug is online. The plug sends a 1010 datagram packet every 20 min periodically to keep the server informed of the online status of the plug.

Step 4: Upon receiving the messages with the command type 1010 from the plug, the authentication server sends a response UDP packet. The command type of this response packet is "1020." This packet contains the smart plug's MAC address (sent to the server in step 3) and the status value.

B. Authentication Phase

There are two different scenarios in the authentication phase. In the first scenario, the plug and the controller are located in different networks. In the other scenario, they are in the same WLAN. We elaborate the communication procedure of these scenario.

1) Plug and Controller in Different Networks: Fig. 3 illustrates the authentication procedure between the smart plug and the controller that are in different networks. For example, the smart plug is located in the home network and connects to the home WLAN while the controller accesses the Internet through the cellular network or another WLAN. In this case, the smart plug authenticates the controller via the authentication server.

Step 5a: The controller sends a UDP request with a command type of "1030" to the authentication server. The UDP port of the authentication server for the controller is 8766. The request packet contains a credential in the "auth value" field for authentication and information of the MAC address.

connect to a same local network. Once the tool is opened, it performs the operation in step 5b to determine if the plug and the controller are in the same WLAN. The plug performs step 6b to send the information of the plug to the tool. After receiving the plug information, the tool displays the plug model, MAC address, IP address, firmware version, and the upgrade status. If a new version is available, the upgrade status shows that a new firmware version can be used. The user can click the new version on the tool, which pops up a prompt box and asks the user to input the password of the plug. After gaining the password from the user, the firmware upgrade tool generates a firmware (i.e., a Linux *bin* file) in a temp file folder and then uploads this firmware to the HTTP server on this plug. The password is encoded in the HTTP header using the HTTP basic authentication method. The plug installs this firmware after receiving the file and restarts. Once this firmware upgrade process is completed, the plug automatically connects to the AP and the user can use the controller to access the plug again.

IV. Security Vulnerabilities of Smart Plug

In this section, we introduce four attacks, i.e., device scanning attack, brute force attack, device spoofing attack, and firmware attack in detail. We also discuss the possible impact after an attacker can access the plugs. Please note that we use our own smart plugs for security analysis in order to avoid legal issues.

A. Device Scanning Attack

In a device scanning attack, the attacker can scan all plugs by enumerating possible MAC addresses of the smart plugs from this vendor. According to recent research [3], many users do not change the default password after deploying their IoT devices. They expect the vendor takes care of the security. Recall that in the authentication phase between the plug and the controller, the controller can receive the 1070 packet as discussed in step 8a if the plug is online and the password is correct. An attacker can craft an authentication message that specifies the plug MAC address, the default username and password, i.e., *admin : 1234*, and check if any victim is using a plug with the specified MAC and the default password. Here "admin" is hard coded and actually does not play the role as a username since the username is not used to differentiate different controllers or users. The MAC address of the plug works as kind of username.

The key to a successful device scanning attack is to know the MAC address space of the smart plug. Luckily for the attacker (unluckily for the manufacturer), MAC addresses are predictable. We can search the MAC address spaces allocated to a company/manufacturer on the Internet [6]. The first six digits of an MAC address indicate the device manufacturer and the other six digits refer to a specific MAC address given to the manufacturer. A manufacturer often gives a block of sequential MAC addresses to the same product. Therefore, if we buy a few smart plugs, we can guess at least portions of MAC addresses allocated to smart plugs of this model. The attacker can enumerate the whole MAC address space of a manufacturer in a brute force attack.

TABLE I
Response to Controller That Sends Authentication Messages to Plug

	Password Correct	Password Wrong
Plug Online	1070	no response
Plug Offline or N/A	5000	5000

Table I shows the possible responses to a controller that sends an authentication message to a plug with a specified MAC address and password. If the plug with the specified MAC address is online and the password is correct, the adversarial controller can receive the 1070 packet. If the plug with the specified MAC address is online and the password is wrong, the plug sends a packet with a command type of 1120 to the authentication server and the authentication server will not forward the message to the controller. To deal with this case in programming, the attacker should set a timer and try more times if the attacker does not obtain a response packet in case that the 1070 UDP packet is lost during the transmission. If the plug with the specified MAC address is offline or does not exist, the authentication server sends a 5000 packet to the attacker. Therefore, when a 5000 packet is received, the attacker cannot tell if the plug with the specified MAC address is offline or there is no plug with that MAC address. However, this does not affect the device scanning attack. Based on Table I, the attacker can leverage the server response in order to find plugs with default passwords and plugs not using the default password/specified password.

B. Brute Force Attack

After deploying the scanning attack, the attacker can discover all the online plugs using nondefault passwords. Then the attacker can select those plugs, construct 1030 packets, and enumerate all possible passwords. The attacker just needs to wait until she receives the right response. At the time of the writing, our experiments show that the authentication server does not block this brute force password attack.

However, our experiments show that the Edimax plug system actually allows a password of 20 characters, including digits and upper- and lower-case alphabetic letters. This password policy is not written in any of the provided manual and we cannot find it online either. If a user indeed inputs a long and complicated password, the brute force attack does not work anymore. Unluckily, the plug system suffers from the following device spoofing attack, which can expose any plug credential.

C. Device Spoofing Attack

1) Attack Process: In the device spoofing attack, we create a software bot that mimics a plug and performs the authentication with the remote controller in order to directly obtain the credential from the controller. It works as follows.

1) The attacker first selects a target plug with a specific MAC address. Recall the attacker knows this plug is online and this plug does not use the default password by using the device scanning attack. If the attacker has

sufficient resources, she can simultaneously choose as many targets as she wants.

2) The attacker registers the spoofed plug by performing step 3 in Section IV. In particular, the attacker can emulate the communication behavior of a real plug and send a packet with a command type of 1010 to the authentication server. Since the server does not provide any authentication method to authenticate the plug, it reg-

system and perform further attacks, e.g., ware into the plug. In this attack, we assume t access the local network of the plug and monitor t fic between plug and controller so as to derive the encoded WiFi username and password in the HTTP header as presented in step 7b. The attacker can then leverage the username and password for the authentication purpose and upload the malicious firmware to the HTTP server on the plug as illustrated in Section III-D.

The attacker is capable of modifying the firmware in order to add the malicious functionality since the vendor of the smart plug provides the open sour find that s the light wei of the plug, is prebui tion protocol and functionalities we find that *BusyBox* is used to provi tools and its source code is available. As a re reconfigure *BusyBox* to enable *Netcat*. To establish a re tunnel to the attacker's desired server, she can utilize a *Netcat* command like *nc* [*IP address*] [*port*]-*e/bin/sh*, where the IP address and the port are those of the attacker's remote server.

The attacker can embed a piece of malicious code into the source code of the DHCP service so that the DHCP can execute the malicious command of *Netcat* at startup. We find that the system of the plug uses the DHCP service provided by BusyBox to assign an IP address to the associated controller. Consequently, the attacker can modify the source code of the DHCP service to add the malicious code and then recompile the source code of the firmware. In this way, the attacker can have a customized firmware and upload it to the HTTP server of the plug. The plug will automatically install the malicious firmware and restart the system. The DHCP service automatically starts at the boot up time and executes the malicious command of *Netcat*.

V. Evaluation

We have implemented the four attacks introduced in Section IV and performed real-world experiments to demonstrate the feasibility and effectiveness of the attacks against the Edimax plug of interest. In this section, we first introduce the experiment setup and then present the experimental results.

A. Experiment Setup

We deployed five plugs and connected them to the Internet via wireless routers. Three plugs were deployed on a university campus in North America while the rest were deployed in Asia. iPads are installed with the plug control app and are used as the controller. Our attack program was implemented in Python.

B. Experimental Results

We first test the scanning attack on the five plugs. Two plugs use the default password, i.e., 1234, two plugs use nondefault passwords, and the fifth plug is not connected to the Internet.

VI. Defense Strategies

In this section, we present guidelines of the defense strategies to mitigate the risks from the Edimax plug vulnerabilities exposed in this paper.

A. Secure Communication Protocol

Cryptography has to be employed to encrypt communication. Encoding and obfuscation are not enough to provide secret communication. In this paper, we can see that an attacker can crack the obfuscation algorithm by analyzing the network traffic. With an eavesdropping attack, she can observe all the plaintext transmitted between the plugs and the controller. To mitigate these threats, secure communication protocols should be adopted, e.g., DTLS, TLS/SSL, and HTTPS, to encrypt the content transmitted between the plug, the controller, the authentication server, and the command relay server.

B. Mutual Authentication Between Plugs and Servers

The spoofing attack stems from the fact that the authentication server does not authenticate the genuine plug. The attacker only needs to send a legitimate datagram using a command type of 1010 and the MAC address of the victim's plug in order to fool the authentication server. The device authentication mechanism should be adopted at both server side and plug side. For example, the device vendor can assign a public/private key pair to a device before it leaves the factory. The authentication server hosts a database of public keys of all the plugs. Therefore, the authentication server can adopt the public-key authentication to authenticate the genuine devices.

There is a possibility of spoofed server attacks against the authentication server and relay server. An attacker may employ DNS poisoning or man-in-the-middle attacks and pretend to be the two servers. To counter the attack, plugs and control apps should be preinstalled with public keys of the two servers and verity certificates of the two servers before transmitting authentication credentials and data.

C. Intrusion Detection System

To thwart the scanning attack, an intrusion detection system should be employed at the server side. The intrusion detection system should be able to identify extensive scanning attacks. For example, it should detect the continuous and ra(v)15.7(e).3(rity)-41-419.6(the)-418erv

and privacy. Dhanjani [19] hacked the Phillips Hue lightbulb

[21] M. Rahman, B. Carbunar, and M. Banik, "Fit and vulnerable: Attacks and defenses for a health monitoring device," in *Proc. 6th Workshop Hot Topics Privacy Enhancing Technol. (HotPETs)*, 2013, pp. 1–12.
[22] J. Obermaier and M. Hutle, "Analyzing the security and privacy of cloud-based video surveillance systems," in *Proc. 2nd ACM Int. Workshop IoT Privacy Trust Security (IoTPTS)*, Xi'an, China, 2016, pp. 22–28.
[23] H. Li *et al.*, "Demographics inference through Wi-Fi network traffic analysis," in *Proc. 35th IEEE Int. Conf. Comput. Commun. (INFOCOM)*, San Francisco, CA, USA, 2016, pp. 1–9.
[24] C. Zuo, W. Wang, R. Wang, and Z. Lin, "Automatic forgery of cryptographically consistent messages to identify security vulnerabilities in mobile services," in *Proc. Netw. Distrib. Syst. Security Symp. (NDSS)*, 2016, pp. 1–17.
[25] F. Maggi, A. Volpatto, S. Gasparini, G. Boracchi, and S. Zanero, "A fast eavesdropping attack against touchscreens," in *Proc. 7th Int. Conf. Inf. Assurance Security (IAS)*, 2011, pp. 320–325.
[26] Q. Yue *et al.*, "Blind recognition of touched keys on mobile devices," in *Proc. 21st ACM Conf. Comput. Commun. Security (CCS)*, Scottsdale, AZ, USA, 2014, pp. 1403–1414.
[27] A. J. Aviv, K. Gibson, E. Mossop, M. Blaze, and J. M. Smith, "Smudge attacks on smartphone touch screens," in *Proc. Workshop Offensive Technol. (WOOT)*, Washington, DC, USA, 2010, pp. 1–7.
[28] Y. Zhan *et al.*, "Fingerprint attack against touch-enabled devices," in *Proc. 2nd Workshop Security Privacy Smartphones Mobile Devices (SPSM)*, Raleigh, NC, USA, 2012, pp. 57–68.
[29] R. Romana, J. Zhou, and J. Lopez, "On the features and challenges of security and privacy in distributed Internet of Things," *Comput. Netw.*, vol. 57, no. 10, pp. 2266–2279, Jul. 2013.
[30] J. S. Kumar and D. R. Patel, "A survey on Internet of Things: Security and privacy issues," *Int. J. Comput. Appl.*, vol. 90, no. 11, pp. 20–26, 2014.
[31] S. Sicari, A. Rizzardi, L. A. Grieco, and A. Coen-Porisini, "Security, privacy and trust in Internet of Things: The road ahead," *Comput. Netw.*, vol. 76, pp. 146–164, Jnn. 2015.
[32] H. Song, G. A. Fink, and S. Jeschke, *Security and Privacy in Cyber-Physical Systems: Foundations, Principles and Applications*. Chichester, U.K.: Wiley, 2017.
[33] J. Noorman *et al.*, "Sancus: Low-cost trustworthy extensible networked devices with a zero-software trusted computing base," in *Proc. USENIX Conf. Security*, Washington, DC, USA, 2013, pp. 479–494.
[34] P. Koeberl, S. Schulz, A.-R. Sadeghi, and V. Varadharajan, "TrustLite: A security architecture for tiny embedded devices," in *Proc. Eur. Conf. Comput. Syst. (EuroSys)*, Amsterdam, The Netherlands, 2014, Art. no. 10.
[35] F. Brasser, B. E. Mahjoub, A.-R. Sadeghi, C. Wachsmann, and P. Koeberl, "TyTAN: Tiny trust anchor for tiny devices," in *Proc. Design Autom. Conf. (DAC)*, San Francisco, CA, USA, 2015, pp. 1–6.
[36] K. Eldefrawy, A. Francillon, D. Perito, and G. Tsudik, "Smart: Secure and minimal architecture for (establishing a dynamic) root of trust," in *Proc. Netw. Distrib. Syst. Security Symp. (NDSS)*, 2012, pp. 1–15.
[37] A. Seshadri *et al.*, "Pioneer: Verifying code integrity and enforcing untampered code execution on legacy systems," in *Proc. ACM Symp. Oper. Syst. Principles (SOSP)*, Brighton, U.K., 2005, pp. 1–16.
[38] A. Seshadri, M. Luk, A. Perrig, L. van Doorn, and P. Khosla, "SCUBA: Secure code update by attestation in sensor networks," in *Proc. ACM Workshop Wireless Security (WiSec)*, Los Angeles, CA, USA, 2006, pp. 85–94.
[39] A. Bates, R. Leonard, H. Pruse, K. R. Butler, and D. Lowd, "Leveraging USB to establish host identity using commodity devices," in *Proc. Netw. Distrib. Syst. Security Symp. (NDSS)*, 2014, pp. 1–14.
[40] D. Formby, P. Srinivasan, A. Leonard, J. Rogers, and R. Beyah, "Who's in control of your control system? Device fingerprinting for cyber-physical systems," in *Proc. Netw. Distrib. Syst. Security Symp. (NDSS)*, 2016, pp. 1–15.

Zhen Ling received the B.S. degree from the Nanjing Institute of Technology, Nanjing, China, in 2005, and the Ph.D. degree from Southeast University, Nanjing, in 2014, both in computer science.

He is an Assistant Professor with the School of Computer Science and Engineering, Southeast University. His current research interests include network security, privacy, and Internet of Things.

Dr. Ling was a recipient of the ACM China Doctoral Dissertation Award in 2014, and the China Computer Federation Doctoral Dissertation Award in 2015.

Junzhou Luo received the B.S. degree in applied mathematics and the M.S. and Ph.D. degrees in computer network from Southeast University, Nanjing, China, in 1982, 1992, and 2000, respectively.

He is a Full Professor with the School of Computer Science and Engineering, Southeast University. His current research interests include next generation network architecture, network security, cloud computing, and wireless LAN.

Dr. Luo is a member of the IEEE Computer Society and ACM, the Co-Chair of the IEEE SMC Technical Committee on Computer Supported Cooperative Work in Design, and the Chair of ACM SIGCOMM China.

Yiling Xu received the B.S. degree in digital media technology from Jiangnan University, Wuxi, China, in 2016. She is currently pursuing the M.S. degree in computer science and engineering with Southeast University, Nanjing, China.

Her current research interests include Internet of Things, privacy, and security.

Chao Gao received the B.S. degree in electrical engineering from Xi'an Jiaotong University, Xi'an, China, in 2011, and the M.S. degree in electrical and computer engineering from Northeastern University, Boston, MA, USA, in 2015. She is currently pursuing the Ph.D. degree in computer science at the University of Massachusetts at Lowell, Lowell, MA, USA.

Her current research interests include Internet of Things and network security and privacy.

Kui Wu (SM'07) received the B.Sc. and M.Sc. degrees in computer science from Wuhan University, Wuhan, China, in 1990 and 1993, respectively, and the Ph.D. degree in computing science from the University of Alberta, Edmonton, AB, Canada, in 2002.

He joined the Department of Computer Science, University of Victoria, Victoria, BC, Canada, in 2002, where he is currently a Professor. His current research interests include network performance analysis, online social networks, Internet of Things, and parallel and distributed algorithms.

Xinwen Fu received the B.S. degree from the University of Science and Technology of China, Hefei, China, in 1995, the M.S. degree in electrical engineering from Xi'an Jiaotong University, Xi'an, China, in 1998, and the Ph.D. degree in computer engineering from Texas A&M University, College Station, TX, USA, in 2005.

He is an Associate Professor with the Department of Computer Science, University of Massachusetts at Lowell, Lowell, CA, USA. His current research interests include network security and privacy, digital forensics, wireless networks, and network QoS. His research was reported by various media such as Wired and aired on CNN and CCTV 10.