Security Analysis of Mobile Device-to-Device Network Applications

Kecheng Liu, *Student Member, IEEE*, Wenlong Shen, *Member, IEEE*, Yu Cheng, *Senior Member, IEEE*, Lin X. Cai, *Member, IEEE*, Qing Li, *Member, IEEE*, Sheng Zhou, *Member, IEEE*, and Zhisheng Niu, *Fellow, IEEE*

***Abstract*—Mobile device-to-device (D2D) network has now become a standardized feature in many mobile devices, by which mobile devices can communicate with each other even when commercial Internet access is not available. Because D2D network is expected to be an intrinsic part of the Internet of Things (IoT) and mobile device is the smartest and the most advanced commercial device in everyday usage, the D2D feature and related security protocols it adopts influences the design and implementation of many other IoT devices. While D2D network provides tangible benefits to users, it also raises the security risks of information leaking. This paper presents an in-depth empirical security analysis on mobile D2D network among Android devices. Android apps could establish a mobile D2D network in various ways, including Wi-Fi hotspot, Wi-Fi Direct, and Bluetooth. Those mobile D2D protocols normally take different protection mechanisms, which makes security investigation considerably challenging. In this paper, we focus on most popular apps in the Google Play Store, with aggregated downloads more than 500 million. Our analysis reveals some critical vulnerabilities. The key findings are bi-fold. First, the current mobile D2D network framework enabled by Android has significant flaw of overprivilege issue. Second, we have identified that most data transfer over mobile D2D network is unencrypted. Furthermore, we exploit the identified Android framework flaws to construct three proof-of-concept attacks and we conclude this paper with security lessons and suggestions of possible solutions against the identified security issues.**

***Index Terms*—Android, device-to-device (D2D), information security, Internet of Things (IoT), overprivilege.**

I. Introduction

MOBILE device-to-device (D2D) network provides a paradigm to facilitate data exchange among physically proximate devices. As an alternative network structure to infrastructure-based network, a mobile D2D network can be established without the presence of base stations or access point. Data transfer takes place directly among connected devices without any interaction with other devices outside the network [2], [8].

Relying on the infrastructure-based network is not always the optimum route to transmit and receive data, especially considering the rapid development of Internet of Things (IoT). In an IoT ecosystem, devices initiate communications among themselves. Due to constrained battery life, processing power, memory, and uncertainty in wireless environment, building an IoT system that solely relies on cloud network is extremely challenging and impractical. The fundamental requirement of the IoT is to provide connectivity between devices. Thus, achieving a seamless end-to-end D2D communication is imperative for the success of the IoT. The rapid growth of mobile devices has made an inception of its leading role in terms of Things (devices). They have been optimized in all aspects, including power, usability, and networking, and have become a significant part of the IoT ecosystem. For both the compatibility and security reasons, many IoT devices have adapted the D2D network feature from mobile devices and have followed their design mechanisms [4]. Mobile D2D network may enable applications over free spectrum, e.g., WiFi band, without involving commercial Internet access [18]. D2D network can also be utilized to offload cellular data traffic for new applications, such as data sharing, multiplayer gaming, and video streaming [9], [24]. In this paper, we particularly focus on mobile D2D network enabled by Wi-Fi-based technologies. Such D2D network is infrastructureless, self-formed networks, and allow free usage without payment charge. In fact, the post popular mobile D2D network apps currently available on market are over such WiFi-based D2D network.

Our group performed, to our best knowledge, the first security analysis of the mobile D2D network framework on Android. Specifically, we empirically evaluated the security design of the most popular smart phone platform for mobile D2D network on Android. We focused on the framework of the network driver and the software implementation of apps because they are the substrates that unify application, protocols, and devices to realize mobile D2D network benefits.

Our Contributions: We discovered security-critical design flaws in the following areas.

Manuscript received March 23, 2018; revised June 24, 2018 and September 26, 2018; accepted October 11, 2018. Date of publication October 22, 2018; date of current version May 8, 2019. This work was supported in part by the NSF under Grant ECCS-1610874, Grant ECCS-1554576, and Grant CNS-1816908, in part by the National Natural Science Foundation of China under Grant 61628107, Grant 61861136003, Grant 61571265, Grant 91638204, and Grant 61621091, and in part by the United Technologies Research Center under Grant A18-0056-001. *(Corresponding author: Yu Cheng.)*

K. Liu, W. Shen, Y. Cheng, and L. X. Cai are with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL 60616 USA (e-mail: kliu6@hawk.iit.edu; wshen7@hawk.iit.edu; cheng@iit.edu; lincai@iit.edu).

Q. Li is with the Network Protection Products Business Unit, Symantec Cooperation, Mountain View, CA 94043 USA (e-mail: qing_li@symantec.com).

S. Zhou and Z. Niu are with the Tsinghua National Laboratory for Information Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: sheng.zhou@tsinghua.edu.cn; niuzhs@tsinghua.edu.cn).

Digital Object Identifier 10.1109/JIOT.2018.2877174

1) Android D2D network needs implicit configurations to make sure the network is safe. The open-sourced API does not present such configuration.
2) Android D2D network framework has significant over-privilege issues. Granted permission to access Wi-Fi states, any apps installed on the smart phone could access the Wi-Fi hotspot feature and the Wi-Fi Direct feature on the device, even if those features are not required by that app. As a result, key information, including password, about the D2D network is exposed, leaving the network unprotected.
3) Once received the key information about the D2D network, any device could hack into the network silently since the network authentication mechanism is evitable. An attacker could join the D2D network without any authentication process. The peers in the network may not even notice.
4) Coarse security implementation of D2D apps leaves data unencrypted. Attackers could eavesdrop on the network and steal information.

We exploited an aggregation of framework design flaws to show various security problems conspired to weaken mobile D2D network security. We constructed three proof-of-concept attacks.

1) We exploited an exiting mobile D2D app available on the app store to snoop network information. Nearby devices running the same hacking program could silently join the network.
2) We eavesdropped on the D2D network and once a file transfer service is initiated, the hacking program can steal the file.
3) We captured and investigated the unencrypted D2D network packets on another existing mobile D2D app, which disclose the key information during a file transfer.

Finally, in our forward looking analysis, we distilled the key studies to establishing secure mobile D2D network framework. We couple the lessons with a breakdown of pros and cons of the tradeoffs in constructing such framework. Our analysis suggests that most problems are readily solvable.

The remainder of this paper is organized as follows. Section II introduces the mobile D2D network architecture and the threat model. Section III presents our analysis on the potential attacks that the D2D network is facing. In Section IV we analyze four most popular D2D applications and show that they suffer from the framework design flaws. We presents three proof-of-concept attacks with experimental results in Section V, followed by the improvement suggestions in Section VI. Section VII reviews the related work and Section VIII concludes this paper.

II. Mobile D2D Network Architecture and Threat Model

A. Wi-Fi Hotspot

Wi-Fi hotspot, also known as Wi-Fi Tethering, has been gaining popularity as a convenient, on-the-move, and cost-effective wireless Internet access technology. A Wi-Fi hotspot network is a star-like cluster-based network with a central node providing 3G/4G data tethering and allowing plural surrounding nodes to join and to gain Internet access. In this paper we will henceforth refer to the central node and the surrounding node as the cluster head (CH) and the cluster member (CM).

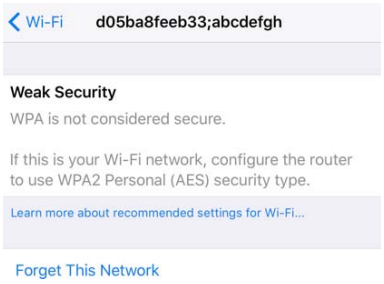


Fig. 1. IOS recognizes the hotspot network created by an Android app utilizing WPA_PSK protection mechanism and alerts the user the security type is not considered secure.

In Android system, developers can access Wi-Fi hotspot feature programmatically. This is how they use it to implement the D2D feature in apps. However, due to Android D2D framework restrictions, programming D2D implementation through API may cause issues from three aspects.

1) Wi-Fi Tethering is System API: To turn on/off the Wi-Fi hotspot feature programmatically, the developer must have an open-sourced API to control the hotspot driver. However, the Wi-Fi hotspot enabling method is a system API. In order to access this API, the developer must use reflection in Java to invoke this method using the method's name. However, using reflection is problematic because it undermines the runtime certainty. More specifically, the Java compiler cannot check on reflection. The reflected code may compile but may explode at runtime because reflection uses the name for the method rather than referring to the actual method. The method to enable/disable Wi-Fi hotspot can be invoked through reflection but it subverts the strength of Java.

2) WPA2_PSK is System API: To create a WPA2_PSK secured hotspot network programmatically, the developer must set the Wi-Fi configuration parameter to support WPA2_PSK key management. However, the WPA2_PSK is a system API as well; developers cannot access this security setting directly. Even though Java reflection also works on WPA2_PSK, many developers neglect the importance of network security and leave the network vulnerable. We found out several high download apps insecure due to coarse-grained software implementation. They use WPA_PSK as an alternative security setting to protect the hotspot network. Many commercial devices today recognize WPA_PSK as an insecure protection. Fig. 1 shows the Wi-Fi interface of an IOS device connecting to a hotspot network created by an Android app. The IOS device recognizes that the network utilizes WPA_PSK to encrypt data and it alerts the user the network is insecure.

3) Hotspot Without Password Absolutely Unprotected: The use of hotspot imposes the risk of people capturing real-time traffic over the wireless connections. Attackers can easily capture, from the air, the packets of unsecured connections to hotspots. It is intuitive that open hotspot has no security protection but many hundred million download apps still

TABLE I
COMPARISON BETWEEN WI-FI DIRECT AND WI-FI HOTSPOT ON ANDROID PLATFORM

	Wi-Fi hotspot	Wi-Fi Direct
Data rate	54Mbps	
Range	100m line-of-sight	
Topology	Cluster based or star-like	
Internet tethering	yes	no
Connection	legacy	D2D and legacy
Security	WEP, WPA_PSK, WPA2_PSK	WPA2_PSK by default
Pairing	no	yes
Configurable SSID and password	yes	no
Wi-Fi client		
Connectionless service broadcast	no	yes
System API	yes	no

choose to use open hotspot, seeking to maximize the convenience. Those apps leverage the user experience over security but expose the information of the smart phone to the public.

B. Wi-Fi Direct

The essential functionality of Wi-Fi Direct is to enable Wi-Fi devices to connect directly without joining a central AP. Wi-Fi Direct is integrated into Android as Wi-Fi Direct and the API is standardized by Android since Android 14. Wi-Fi Direct is a software realization sharing the same network interface card with the Wi-Fi module. Comparing with hotspot, there are many similarities and differences. We summarize them in Table I.

Wi-Fi

1) Wi-Fi Direct Versus Wi-Fi Hotspot: Direct inherits the high-speed advantages from 802.11. The data transfer rate can reach 54 Mb/s. Since both Wi-Fi Direct and hotspot uses the same hardware, the RF coverage is 100 line-of-sight.

Similar to a hotspot, the Wi-Fi Direct network is also a star-like, cluster-based network with a central node acting as a CH and surrounding nodes as CMs. In contrast with hotspot, the Wi-Fi Direct cluster is defined as a group in the Wi-Fi Alliance standard. The CH is defined as the group owner (GO) and the CM is defined as the group member (GM).

Like a Wi-Fi AP, a GO is visible by other Wi-Fi devices on the Wi-Fi scan list; therefore, GO is sometimes called a soft AP. Nevertheless, since Wi-Fi Direct is designed to provide the D2D feature so the GO does not provide the Internet tethering function to the GM.

2) Two Connection Types: There are two ways to join a Wi-Fi Direct group: 1) through Wi-Fi Direct connection and 2) through legacy connection. A wireless link connecting a device to a GO following the Wi-Fi Direct protocol is called a D2D connection. On the other hand, any Wi-Fi device, regardless of Wi-Fi Direct support, can associate to the GO following the Wi-Fi protocol. The associated client device is defined as a legacy client and the corresponding connection is called a legacy connection.

3) Different Security Mechanisms: Wi-Fi Direct also acquires all the security mechanism from 802.11. A Wi-Fi Direct network is by default protected by WPA2_PSK. As discussed previously, a GO is a soft AP which can be connected through a legacy connection. Once created, the soft AP is WPA2_PSK encrypted and the password is randomly generated by the network driver.

Additionally, if the group is established through the Wi-Fi Direct protocol, extra protection mechanisms are introduced in the standard. A human-involved paring authentication process, which is a software transform of Wi-Fi protection settings (WPS), is integrated into the Android network driver. Specifically, there are three WPS options: 1) display pin configuration; 2) keypad pin configuration; and 3) push button configuration. Since the two-way authentication is a requirement by the Wi-Fi Direct standard, it is an inevitable procedure during a D2D connection using Android devices.

4) Service Broadcast and Discovery: One of the most important features of Wi-Fi Direct is connectionless service broadcast and discovery (SBD). Using SBD allows devices to discover the services of nearby devices directly, without being connected to a network. A device can also advertise the services running on it. These capabilities help devices communicate between apps, even when no local network is available.

C. Threat Model

Our effort concentrated on systematically discovering and exploiting mobile D2D network framework design vulnerabilities. Any scheme relevant to the above aspects were within scope. We did not study attacks that attempt to crack the security algorithm of the D2D network with advanced hacking tools. Bugs in those areas were discussed and fixed in 802.11 security analysis. In contrast, attacks focused on software design flaws had extensive influences since programming frameworks were burdensome to change without significant disruption once there was a large set of applications that used the framework.

We focused this paper on D2D communication and related applications which are designed independent of Internet availability. We aimed to create an offline attacking scenario regardless whether Android devices, not limited to smartphones, have Internet access or not. The key point we are trying to demonstrate is despite Android implements the D2D features following the well-defined network standards and it includes the well-known security mechanism, the D2D network it creates has a backdoor to the intruders.

D. Conclusions of the Section

In this section, we introduce the background and the architectural design of two mobile D2D network protocols based on the 802.11 standard: Wi-Fi hotspot and Wi-Fi Direct. By making comparison and contrast, we claim that Wi-Fi Direct is better implemented and protected on Android platform. The most severe issue of Wi-Fi hotspot is the absence of open-sourced API and the inadequacy of appropriate security configurations. Android does not present an implicit setting for developers to securely invoke the Wi-Fi hotspot feature. Nevertheless, many apps still prefer to using it as the D2D network basis due to its convenience rather than Wi-Fi Direct.

In later sections, we will illustrate our security analysis of the mobile D2D network framework. Many of our

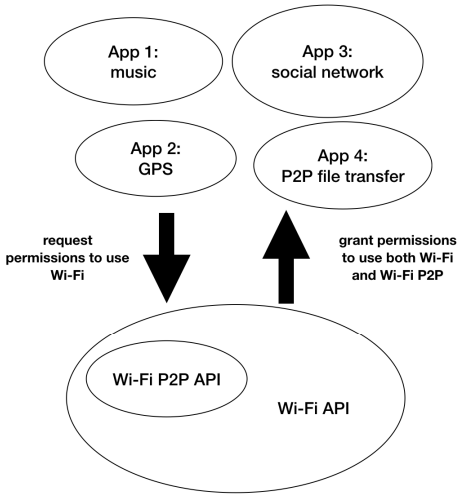


Fig. 2. Over privilege issue in Android network framework caused by permission management.

investigations depend on the details emphasized on this section.

III. Security Analysis of Mobile D2D Network Framework

We break down the security of the Mobile D2D network framework into four general themes. Our methodology involves creating a list of attack models based on this paper of the mobile D2D network architecture and extensively testing each potential attack model with prototype mobile D2D apps. We describe each investigation and will expound our findings later in this section.

A. Overprivileged D2D Network Access

Android uses permission to avoid overprivilege issues. To maintain security for the system and users, Android requires apps to request permission before the apps can use certain system data and features. Depending on how sensitive the area is, the system may grant the permission automatically, or it may ask the user to approve the request. Thus, if an application does not request to use a certain feature but the permission to access the feature is still granted, this accidental permission acquisition is an example of overprivilege.

As shown in Fig. 2, Android does not provide any permission to particularize the accesses to Wi-Fi hotspot and Wi-Fi Direct; which means to establish an 802.11-based D2D network, an application only has to acquire the Wi-Fi permissions. This framework design articulates the overprivilege problems because any app that has access to the Wi-Fi feature will have the privilege to utilize the D2D network feature as well.

The permissions to use Wi-Fi is not a sensitive permission for users because most applications installed on the smart phone need to have the access to use Wi-Fi and the Internet, such as the music app, the GPS app and the social network app. Users will not pay extra attention to an app that requests the usage of Wi-Fi if they believe that application needs to use the Internet.

This framework design leaves applications that rely on the D2D feature in significant danger because there is only one network driver on the Android device and the information about the network is singleton; which means the SSID and the password are consistent on the framework level regardless which app is trying to get it. The overprivilege issue allows almost every app on the device to more or less access the key information to the D2D network. It gives attackers the opportunity to maliciously manipulate the network, including intercept data transmission, eavesdropping, sending false signals etc.

In order to take advantage of the framework design flaw, all attackers need to do is to install a benign-but-malicious app that requests the permission to access Wi-Fi. The framework would ask the user to authorize the permission. Unfortunately, the request is too common so that it will neither alert the user nor the system; therefore, the user is very unlikely to deny the request, making the attack an essentially effortless assignment.

Once the benign-but-malicious app is installed on the smart phone, the sensitive network information is at risk and an offline leakage attack can be executed by the third party. We will discuss this attack model in the following paragraphs.

B. Sensitive Network Information Leakage

As discussed in Section IV-A, the overprivilege issue due to coarse-grained framework opens a back door to a third party app to access key information about the network. Because the framework does not place any restrictions on outbound Internet communication either, such a design flaw allows malicious apps to abuse this ability to leak sensitive information from a victim's smart phone.

However, information leakage via the Internet is not what we concentrate on for two reasons. First, in a D2D network scenario, Internet is assumed unavailable; second, D2D data is transmitted locally without the Internet access. We exploit an offline attack model based on the D2D network architecture described in Section III. Following our attack logic, a third party device can receive key information about the D2D network in offline mode.

Fig. 3 depicts the attack model. Users Alice and Bob both have a D2D app installed on their smart phones and they use the app to establish a D2D network. Because the wireless range is 100 m line-of-sight, Alice and Bob must be in physical proximity to maintain the wireless connection. On Alice and Bob's smart phones, a benign-but-malicious app is also preinstalled which has acquired the Wi-Fi permission from the users. Due to the overprivilege issue discussed in Section IV-A, this app can obtain the key information about the D2D network.

We assume Trudy is the attacker and in order to perform an offline attack, he must locate in physical proximity to Alice and Bob as well. His smartphone must be able to detect the signals transmitted by Alice and Bob's smart phones. This is not an impossible scenario because Trudy can pretend to be a random person standing near Alice and Bob but far enough so that they believe nobody can see their operations on their smart

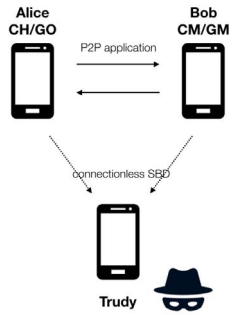


Fig. 3. Offline attack model to obtain key information about the network.

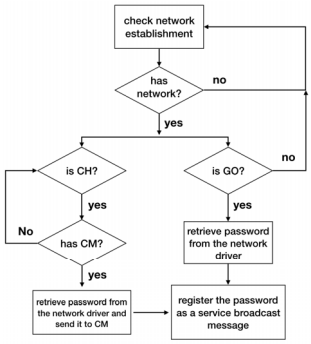


Fig. 4. Flowchart to cause network information leakage due to framework design flaws.

phones. Furthermore, Alice, Bob, and Trudy's smart phones are completely offline in our attack model.

The operational flow chart of the malicious app installed on Alice and Bob's smart phones is shown in Fig. 4. The purpose of the malicious app is to induce network information leakage so the first thing it needs to examine is the D2D network status. If the smart phone is not part of the D2D network, the malicious program maintains its benign behavior. As soon as it detects the network establishment, it will change to the hacking mode. The network establishment event is triggered by multiple Android system intents. We contain all events, including Wi-Fi Direct group creation, D2D connection, Wi-Fi hotspot creation, and Wi-Fi network association, in the same intent receiver.

After the app receives the network establishment event, it identifies which type of the D2D network the smart phone has joined. If the network is a hotspot network, it examines whether the device is a CH or a CM. Similarly, if the network is a Wi-Fi Direct network, it examines whether it is a GO or a GM. We separate the procedures because only CH and GO contain the key information about the network. The D2D application determines either Alice or Bob become the access point of the network, but to Trudy, it does not make any difference which smart phone is the access point. All Trudy anticipates is the network name and its key.

We will first discuss the situation if the smart phone is a GO. Because of the overprivileged framework design, the malicious app now can retrieve the SSID and the password of the Wi-Fi Direct network. Android provides the open-sourced API to achieve this. After the key information has been stolen, the malicious app will take advantage of the Wi-Fi Direct SBD feature. Refer to Section III-B, SBD is a connectionless data transmission feature provided by Wi-Fi Direct. Since the malicious app has access to the entire Wi-Fi Direct API due to overprivilege, it can utilize SBD to accomplish the offline attack. The malicious app can register the SSID and the password as a broadcast service. Any nearby device can discover this service if it scans following the SBD standard. This is exactly what Trudy's smart phone is running. Trudy can program an app on the smart phone to periodically scan for nearby services. Because Trudy is in physical proximity to Alice and Bob, Trudy's smart phone has a great chance to discover the service registered on both Alice and Bob's smart phones.

There is another situation that the D2D network established by Alice and Bob's smart phones is a hotspot network. In this scenario, the malicious app can still retrieve the password of the network due to overprivileged framework design. Although Android does not provide an open-sourced API for the developer to get this information, the developers can still use reflection to invoke a hidden system API called: getWiFiApConfiguration.

The next step of the offline attack is different from the previous situation. Refer to the context in Section III-B, because Wi-Fi hotspot and Wi-Fi are not compatible and Wi-Fi Direct requires the smart phone to enable Wi-Fi, the key information retrieved by the malicious app cannot get registered by SBD on the smart phone which provides hotspot service. To make a successful offline attack, the malicious app can exploit the client smart phone because the client must have the Wi-Fi turned on in order to connect to the hotspot. Thus, after the malicious app has got the key information, it patiently waits for the connection from the client device. Once the client has joined the hotspot network, it transmits the password to the client device and the malicious app installed on the client device will register the password using the SBD feature. Trudy's smart phone still scans for nearby services and eventually, it will discover the password registered on the client device.

The offline key information leakage is now complete: Trudy steals the password that protects the D2D network established by Alice and Bob without their notice. The same attack logic applies on both hotspot network and Wi-Fi Direct network on Android platform.

C. Avoidable Human-Involved Authentication Process

In previous sections, we introduced the human-involved two-way authentication process during mobile D2D network establishment. Human involvement is one of the best solutions to eliminate security risk in D2D network because a human can physically identify the appropriate device and approve the connection request. However, due to coarse-grained framework design, hackers can bypass the two-way authentication process.

From Section IV-B, attackers can offline retrieve the password of the network. Now using the password, any device can join the network without undertaking the two-way authentication.

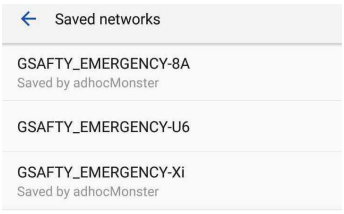


Fig. 5. Example of saved networks by a certain software.

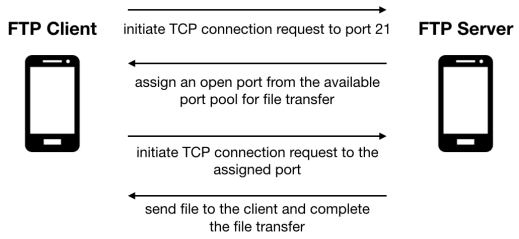


Fig. 6. FTP over D2D network and its work flow.

Wi-Fi Direct network has extra protection mechanism in the standard and Android has integrated the paring procedure on the system level. However, Wi-Fi Direct supports two connection types: 1) D2D connection and legacy connection and 2) human-involved paring only applies to the D2D connection. Because attackers have known the password to the GO, they can join the Wi-Fi Direct network via a legacy connection, which is the same as to join a Wi-Fi AP. In this way, the human-involved paring protection will not get invoked on the Android smart phone.

There is an automated procedure that allows the Trudy's smart phone to join an encrypted Wi-Fi network programmatically. After Trudy's smart phone has received the key information from the previous attack, it follows the attacking operations as described below. The procedure is a combination of network APIs following the Wi-Fi connection logic.

The first step is to create the compatible network configuration of the targeting network. The key points of the new configuration are the SSID, the password and the encryption method. The second step is to save the new configuration to the configuration list in the Android system. Android provides such an API called: addNetwork. The third step is to reconnect to the currently active access point. We use reconnect because the network has been preconfigured and saved in the system. The network driver considers the target network as a past-connected network and it is capable of reconnecting to that access point when nearby. Fig. 5 shows an example of how Android distinguish the programmatically saved networks and the typed-in saved networks. It marks the app's name below the SSID.

Through those three steps, attacker circumvents the human-involved authentication process and joins the D2D network without notifying the owner of the network.

D. Insecure Data Transfer

Developers have great freedom to choose how to transmit data on the D2D network. Android neither specifies nor limits the protocol to make data transfer on the application level. Because file transfer is the most common application using the mobile D2D network, we focus our security analysis on the protocol choices related to file transfers. In the following paragraph we illustrate two popular file transfer protocols on D2D network and their security issues.

1) FTP on D2D Network: FTP can adapt to D2D network and guarantee file transfer service. FTP also applies to a client–server model, but comparing to the direct TCP socket transfer, the server and client roles are inverted. Fig. 6 demonstrates how D2D network utilizes FTP for local service. The two channels, one on port 21 and the other on the feedback port, is an implementation of the FTP out-of-band design. FTP on mobile D2D network inherits the FTP features, such as pause, restart, and continue transfer. It also inherits the FTP log in mechanism. If the password is required by the server, the client has to log in to the FTP server before file transfer. If the password is not required by the server then any device in the network could download the file from the server. FTP does not provide extra encryption mechanism to protect the data transfer.

In Internet scenario, FTP normally combines with secure socket layer (SSL) and secure shell (SSH) to enhance security on transport layer and application layer. In the offline scenario, because the network is assumed private and isolated, SSL and SSH are neglected; even the log in step in FTP is omitted to provide best convenience to users. However, as descripted in previous parts, intruders can exploit the D2D network with a benign-but-malicious app installed on users the cellphone. The D2D network is not an absolute safe network even with strong password protection; therefore, unencrypted data suffers from multiple threats when an intruder hacks into the network. Due to limited access and variety of file transfer apps, we have not identify an app in current market uses the offline FTP model introduced in this part. Nonetheless, we have verified it as a practical D2D file transfer model and examined its possible security vulnerabilities.

2) HTTP on D2D Network: The usage of HTTP on the D2D network is similar to FTP. Fig. 7 illustrates the implementation of HTTP on D2D network. The server side is the smart phone providing data and the client side is the one receiving data. On the server side, it binds port 80, which is the reserved HTTP port, on a server socket and waits for the connection. The client side initiates a TCP connection to port 80. If there is a successful connection established on port 80, then the server device uses HTTP POST to post the file directory on its local server. The server device also updates the post directory to the client device through the same connection. The client device can access the file using HTTP GET. The server places the file packets in the data field of the HTTP GET.

Similar to FTP, HTTP does not provide encryption mechanism to protect data. In the Internet scenario, HTTPS solves the encryption issue. HTTPS is a combination of HTTP on the application layer and SSL on the transport layer. In the D2D network scenario, developers choose HTTP over HTTPS because HTTPS requires certificate authority (CA) but CA can only provide online service. In offline mode,

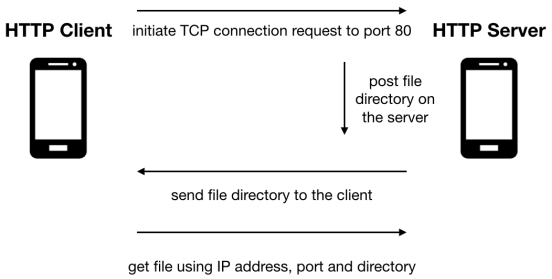


Fig. 7. HTTP over D2D network and its work flow.

TABLE II
STATISTICAL BREAKDOWN OF OUR ANALYSIS

App name	Shareit	Zapaya	Send Anywhere	SuperBeam
Number of Downloads	1 billion	500 million	5-10 million	10-50 million
Wi-Fi Hotspot	Yes	Yes	Yes	No
Wi-Fi Direct	Optional	Optional	No	Yes
Network access protection	Optional	Optional	Required	Required
Protection mechanism	WPA_PSK	WPA_PSK	WPA_PSK	WPA2_PSK
Over Privilege	Yes	Yes	Yes	Yes

CA is unreachable from the smart phone so the protection mechanism may be burdensome. To guarantee the file transfer feature, developer prefers HTTP over HTTPS but sending unencrypted data over HTTP is an insecure application design and Section IV discuss an example attacks targeting on HTTP.

IV. EMPIRICAL SECURITY ANALYSIS OF MOBILE D2D NETWORK APPS

To understand how the security issues discussed in Section IV manifest in practice, we download four highest download file transfer apps from the Google Play Store and performed in depth analysis. The total number of downloads of those apps is over 1.5 billion, according to the statics shown on both Google Play Store and Chinese Android Play Store. Table II contains a statistical breakdown of our analysis on each application. Our analysis shows that all of those apps suffer from the framework design flaws discussed in Section IV.

A. Shareit

Shareit claims to be the number 1 file transfer tool in the world. It has over 500 million users in China, 300 million users in India and over 1 billion users in world wide. Despite being such a popular application, its security implementation is coarse-grained and we found multiple security issues. Before we can make an analysis on the network, we must identify which protocol it selects as the network basis. The default D2D network established by Shareit is via hotspot. Wi-Fi Direct feature is included in the setting and user can enable it. However, Shareit marks Wi-Fi Direct as a Beta feature and it alters users the Beta version is unstable. Thus, we focus our analysis on the hotspot network.

The hotspot network created by Shareit is an open network by default. This is a risky design because anyone can access

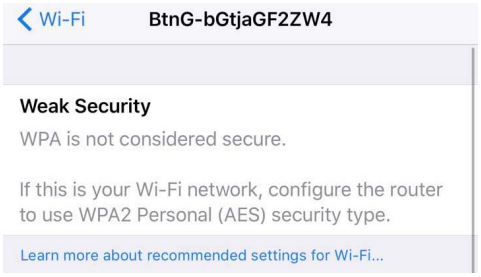


Fig. 8. Shareit creates a WPA_PSK protect network and it is detected as an insecure network.

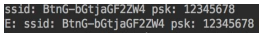


Fig. 9. Shareit experiences the key information leakage problem.

Fig. 10. Context in the file params.txt.

the network regardless of the password. Shareit indeed offers an option for users to setup a customized password but the security level is only WPA_PSK. As we mentioned in Section III, WPA_PSK has been proved not secure. Fig. 8 shows the password protected network created by Shareit is detected as a weak protection network. Fig. 9 shows Shareit experiences key information leakage. A third party benign-but-malicious app can retrieve the same password created by Shareit from the network framework.

On the application layer, Shareit chooses to use HTTP POST and GET as the file transfer protocol instead of HTTPS. This design choice leaves data unencrypted and a packet capture tool can catch the information transmitting on the network. To prove our point, we performed a file transfer using Shareit. The format of the file is .txt and the name of the file is params.txt. This is a just a random .txt file we found in our experiment smart phone. The context of the file is just some random parameters as shown in Fig. 10.

During the file transfer, we started the packet capture tool. Our tool flagged each packet capture when it detected upstream and downstream processed by a certain application. It marked the application and attached the details of the packet afterward. Fig. 11 illustrates the packet capture result. The unencrypted data was exposed to our tool. Any smart phone

z ◆◆◆m; r◆◆{"name":"tianshi","frameRate":15,"additionalInfo":
{"frontTips":{"content":"露个脸吧","isFaceTrack":true}},"beautyFace":
{"bigEyeAmount":0.8,"thinFaceAmount":0.8},"itemList":
[{"layerType":"default","pointIndex":
27,"folder":"guangquan","frames":30,"height":58,"width":
210,"type":0,"offsetX":0,"offsetY":-160,"loopingIndex":
0,"alwaysShow":false},{"layerType":"default","pointIndex":
8,"folder":"xiangquan","frames":10,"height":102,"width":
104,"type":0,"offsetX":3,"offsetY":82,"loopingIndex":
0,"alwaysShow":false},{"layerType":"default","pointIndex":
29,"folder":"lianshi","frames":15,"height":72,"width":
211,"type":0,"offsetX":0,"offsetY":0,"loopingIndex":
0,"alwaysShow":false},{"layerType":"default","pointIndex":
29,"folder":"xingxing","frames":15,"height":76,"width":
322,"type":0,"offsetX":0,"offsetY":0,"loopingIndex":
0,"alwaysShow":false},{"layerType":"default","pointIndex":
24,"folder":"youchibang","frames":7,"height":49,"width":
56,"type":0,"offsetX":42,"offsetY":-55,"loopingIndex":
0,"alwaysShow":false},{"layerType":"default","pointIndex":

Fig. 11. Shareit packet capture result. The unencrypted data is exposed to our tool.

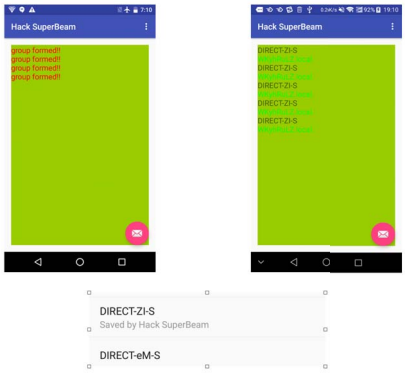


Fig. 12. Top left: Hack SuperBeam catches the group formation event and registers the key information on SBD. Top right: Trudy steals the key information of the network through an offline attack. Bottom: Hack SuperBeam saves the network configuration, allowing Trudy's smart phone to reconnect to Alice's phone.

connected to this network could exploit similar attacks and steal information from this network. We used a .txt file for a better demonstration in our analysis. The attacker can reconstruct any type of file based on the intercepted data and the corresponding file type, which is normally captured in the HTTP header.

B. Zapaya and Send Anywhere

Zapaya and Send Anywhere are very similar tools to Shareit. They also have a large number of users in world wide. According to the statistics on the official website and Google Play Store, they have almost 510 million users globally. We performed the same attack on them and observed the same vulnerabilities as Shareit.

C. SuperBeam

SuperBeam is a Wi-Fi Direct-based D2D file transfer tool. According to the statistics from Google Play Store, it has 1050 million downloads. Refer to the architectural design of Wi-Fi Direct in Section III, the network is automatically encrypted by WPA2_PSK and there is an additional system level human-involved paring protection. Thus, the network created by SuperBeam is essentially safer than the network created by previous three applications. However, due to coarse-grained framework design, SuperBeam also has key information leakage issue and attackers can bypass the paring authentication procedure with a legacy connection.

SuperBeam utilizes HTTP on the application layer to handle file transfer. The implementation of HTTP is slightly different from previous three applications. Other than sending and receiving data exclusively through the application, SuperBeam offers a feature that allows connected devices to download the file from the Web browser. We exploit this feature and construct a new attack model to steal the file from the HTTP server. We will demonstrate such a combinational attack in Section VI.

V. Proof-of-Concept Attacks

We show three concrete ways in which we combine various security design flaws and developer-bugs discussed in Sections III–V to weaken mobile D2D network security. We first present an offline attack that exploits password protected network with a benign-but-malicious app which requests minimum permissions from Android system. We then show an attack that steal the file from a smart phone which intents to establish file transfer service with a peer. In the third attack we show an intruder in the network can eavesdrop on the network and capture data during a file transfer.

A. Offline Network Hacking Attack

The threat model we use in this attack is the same as what we introduced in Section IV-B. We setup three smartphones. Two of them are Alice and Bob and the other one is Trudy. Alice and Bob are trying to establish a D2D network using SuperBeam and Trudy will hack into the network.

On all three smart phones, we installed the benign-but-malicious app and we call it Hack SuperBeam. This application is programmed thoroughly using open-sourced API provided by Android and none of those three smart phones are rooted devices. They are off-the-shell smart phones running the primitive Android operating system.

Hack SuperBeam exploits the overprivileged framework. It seems to have a benign behavior because it does not trigger any sensitive system alarm whatsoever. It requests a few most common permissions from the system as majority application does.

Hack SuperBeam follows the attack logic described in Fig. 4. To snap the network change events, the first step is to register the system actions that reflects the change of the network and connections. The second step Hack SuperBeam takes is to get the key information of the network. Fig. 12 demonstrates Hack SuperBeam receives the group establishment event. After group formation, Hack SuperBeam uses the API to retrieve group information. The third step is to register the captured SSID and password to the connectionless broadcast service using the SBD feature. The final step is to make a legacy connection to the GO. Fig. 12 shows the network configuration is saved by Hack SuperBeam, indicating the final step before connection. Thereafter, Trudy's device can reconnect to Alice's phone with a legacy connection.

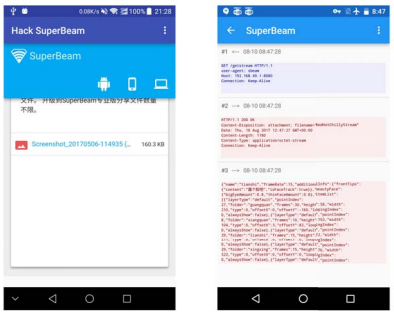


Fig. 13. Left: Trudy can download the image file displayed on his smart phone using Hack SuperBeam. Right: packet capture on SuperBeam reveals the unencrypted data.

B. File Stealing Attack

In the previous section, we exploit an offline network hacking attack. Trudy is now in the D2D network created by Alice and Bob and he can download the file from Alice.

SuperBeam offers a function for the peers to download the file on a Web browser. Trudy is now a peer in the network and Hack SuperBeam can exploit this feature and download the sharing files.

SuperBeam post the context of the sharing files on its local server. The address of the files is 192.168.49.1:8080. At this point, Trudy can download the file and complete the file stealing attack. Fig. 13 demonstrates the *webview* and URL in Hack SuperBeam that allows Trudy to download the file.

C. Packet Eavesdropping

Packet eavesdropping also applies to SuperBeam because it does not encrypt data throughout the file transfer process. After Trudy hacked into the network, he can use a packet capture tool and eavesdrop on the network. The unencrypted data in Fig. 13 is from the .txt file Alice sends to Bob. Trudy can intercept the data and steal the information.

VI. Improvement Suggestions

We discuss some lessons learned from the analysis of the mobile D2D network on Android that we believe to be broadly applicable to mobile D2D network framework design. We also provide some improvement suggestions.

1) Extra Permission to Access D2D Network: Android does not provide dedicate permissions to access D2D network. Any application installed on the device has equivalent access to the D2D network, even though it does not need to use this feature. This causes overprivilege issues and it becomes the fundamental vulnerability to other threats. We suggest Android place dedicated permissions to control the accessibility of D2D network, including Wi-Fi hotspot and Wi-Fi Direct. This can avoid irrelevant third party app retrieving key information about the network.

2) Key Network Information Protection: Key information about the network should have restricted access. If one application is using the D2D feature and the framework should prevent another application access the key information about the network. If both applications want to share the D2D network feature, the user must get altered and authorize this request.

3) Keep Track of the Number of Clients: The host of the network, the CH or the GO, should keep track of the number of clients in the network to prevent intruders. The host device can read its ARP table and determine how many devices are reachable in the network. If there is an unrecognized device joins the ARP table, the host device should pause the D2D service immediately and alert the user.

4) Data Encryption: There are multiple ways to solve this issue. Developers can preinstalled a shared key in the application to encrypt data. The preinstalled key can also support Diffie–Hellman to establish a temporary shared key for encryption. The purpose of the encryption key is to provide a secure offline transportation layer so data transmission using HTTP and FTP can be secure.

The security issue due to framework overprivilege is of fundamental importance. Since it is a middleware security problem, the solution to eliminate the vulnerability requires the effort from both industrial and academic sides. The solution we proposed in this paper is to avoid such security issues on the application level. App developers can take our advice and make corresponding changes to patch the security bugs. The fixes seem to be simple but the problem is not supposed to be in the middleware in the first place. It is not the developer's responsibility to follow a specific setup of APIs to make their application safe. Nonetheless, we pointed out a nonobvious architectural issue in designing a secure mobile D2D network to which all developers must pay attention, and we give out a solution to avoid the design issue.

VII. Related Work

There are a number of pioneering research works studied multiple aspects of D2D communication. Among them, [7] developed an analytical model for analyzing the coverage probability and ergodic rate of users in the cellular network. Cao *et al.* [6] studied the resource allocation problem for minimizing the end-to-end delay for D2D communications in the vehicular network. Liu *et al.* [16] presented a comprehensive survey of available D2D research works, and outlined several open research problems. As to the security aspect, Shen *et al.* [22] proposed a key establishment protocol for initial trust establishment in a WiFi Direct D2D network, Haus *et al.* [11] presented an extensive review of the state-of-the-art solutions for enhancing security and privacy in D2D communication. The review spanned across a variety of D2D prospects, such as network communication, peer discovery, proximity services, and location privacy. Haus *et al.* [11] also provided a detailed discussion on D2D privacy. It summarized and compared the existing solutions according to security and privacy requirements.

Since this paper focuses on IEEE 802.11-based D2D communication, the thorough studies of IEEE 802.11 security protocols are related to this paper. Lashkari *et al.* [15] explained the structure and problems of WPA and WPA2. Adnan *et al.* [1] made a comparison between WPA and WPA2, and correlated the two with respect to performance.

Raju and Nair [21] considered vulnerabilities like open nature of communication channel, lack of confidentiality, weak encryption methods, etc., in a Wi-Fi hotspot and proposed a security protocol that ensured individual confidentiality during the communication.

Although this paper focuses on the security issue in Wi-Fi related D2D systems, our group has also reviewed works related to the mobile D2D system based on Bluetooth technology. Bluetooth related technologies have been widely adapted in wearable devices, such as watches and wristbands. The security issues inherent in Bluetooth are largely due to the process of pairing one device to another. Bluetooth paring is still subject to Man-in-the-Middle attacks, even after the devices are paired. The biggest factor in Bluetooth vulnerabilities is the version of Bluetooth that is being used, and the security of communications between devices is only as strong as the weakest link, i.e., the device with the oldest (weakest) version. Such vulnerabilities include coarse PIN management, unlimited challenge requests, and weak stream cipher [20], [23].

Some groups had studied the security of D2D network from smart phone applications. Bai *et al.* performed a security analysis on Apple's major ZeroConf components and found out it is mostly unprotected. IOS system is a close-sourced environment and this group took the challenge and concluded ZeroConf systems like AirDrop did not have security guaranteed [3], [19].

The security of Android system is under strict scrutiny. The Android operating system uses the permission-based model which allows applications to access user information, system information, device information, and external resources of the smart phone. Misuse of app permissions is one of the main reasons leading to the overprivilege in the mobile D2D network. Studies of Android security issues presented in [10], [12], [13], and [25] provided a systematic study of the Android security architecture. Buhov *et al.* [5], Kawamoto *et al.* [14], and Liu *et al.* [17] focused on the current issues related to misuse of the network protocols in Android such as HTTPS and SSL.

Vulnerabilities in connections between proximate devices have been studied extensively. Haus *et al.* [11] and Xia *et al.* [25] identified three security threats due to: 1) direct wireless connections; 2) mobility of end users; and 3) privacy issues in social applications. The greater the number of devices that adopt D2D communication, the greater the interest of adversaries to attack these networks [23]. According to recent study [19], security and privacy are open issues for D2D.

VIII. Conclusion

We performed an empirical security evaluation of the mobile D2D framework on Android operating system. Analyzing mobile D2D network was challenging because there were multiple mechanisms to establish a D2D network on Android and each of them uses distinct security protections. Additionally, the apps installed on Android were close-sourced individual modules so we did not know how file transfer service was implemented on the application level.

We performed an overprivilege analysis of four most popular D2D file transfer apps to determine how well the mobile D2D framework protects user's information and privacy. The four apps have over 1.5 billion users combined. We discovered: 1) all those apps suffer from the overprivilege issue due to coarse-grained network framework; 2) key information leakage is a prevailing threat; 3) the system-enforced human-involved authentication process is avoidable; and 4) unencrypted data transfer over network is insecure. We combined these design flaws with other vulnerabilities and constructed three proof-of-concept attack to: 1) hack into a D2D network; 2) steal a file from a network; and 3) eavesdropping on a network to capture unencrypted data. At the end of this paper, we summarized the lessons from our analysis and provided improvement suggestions.

References

[1] A. H. Adnan *et al.*, "A comparative study of WLAN security protocols: WPA, WPA2," in *Proc. IEEE Int. Conf. Adv. Elect. Eng. (ICAEE)*, 2015, pp. 165–169.

[2] A. Asadi, Q. Wang, and V. Mancuso, "A survey on device-to-device communication in cellular networks," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 1801–1819, 4th Quart., 2014.

[3] X. Bai *et al.*, "Apple ZeroConf holes: How hackers can steal iPhone photos," *IEEE Security Privacy*, vol. 15, no. 2, pp. 42–49, Mar./Apr. 2017.

[4] O. Bello and S. Zeadally, "Intelligent device-to-device communication in the Internet of Things," *IEEE Syst. J.*, vol. 10, no. 3, pp. 1172–1182, Sep. 2016.

[5] D. Buhov, M. Huber, G. Merzdovnik, E. Weippl, and V. Dimitrova, "Network security challenges in Android applications," in *Proc. IEEE Int. Conf. Availability Rel. Security (ARES)*, 2015, pp. 327–332.

[6] X. Cao, L. Liu, Y. Cheng, L. X. Cai, and C. Sun, "On optimal device-to-device resource allocation for minimizing end-to-end delay in VANETs," *IEEE Trans. Veh. Technol.*, vol. 65, no. 10, pp. 7905–7916, Oct. 2016.

[7] J. Dai, J. Liu, Y. Shi, S. Zhang, and J. Ma, "Analytical modeling of resource allocation in D2D overlaying multihop multichannel uplink cellular networks," *IEEE Trans. Veh. Technol.*, vol. 66, no. 8, pp. 6633–6644, Aug. 2017.

[8] Z. Degui and Y. Geng, "Content distribution mechanism in mobile P2P network," *J. Netw.*, vol. 9, no. 5, p. 1229, 2014.

[9] G. Ding and B. Bhargava, "Peer-to-peer file-sharing over mobile ad hoc networks," in *Proc. IEEE Annu. Conf. Pervasive Comput. Commun. Workshops*, Orlando, FL, USA, 2004, pp. 104–108.

[10] P. Faruki *et al.*, "Android security: A survey of issues, malware penetration, and defenses," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 2, pp. 998–1022, 2nd Quart., 2015.

[11] M. Haus *et al.*, "Security and privacy in device-to-device (D2D) communication: A review," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 2, pp. 1054–1079, 2nd Quart., 2017.

[12] Y. J. Jia, Q. A. Chen, Y. Lin, C. Kong, and Z. M. Mao, "Open doors for bob and mallory: Open port usage in Android apps and security implications," in *Proc. IEEE Eur. Symp. Security Privacy (EuroSP)*, 2017, pp. 190–203.

[13] S. Karthick and S. Binu, "Android security issues and solutions," in *Proc. IEEE Int. Conf. Innov. Mech. Ind. Appl. (ICIMIA)*, 2017, pp. 686–689.

[14] Y. Kawamoto *et al.*, "A feedback control-based crowd dynamics management in IoT system," *IEEE Internet Things J.*, vol. 4, no. 5, pp. 1466–1476, Oct. 2017.

[15] A. H. Lashkari, M. M. S. Danesh, and B. Samadi, "A survey on wireless security protocols (WEP, WPA and WPA2/802.11i)," in *Proc. IEEE Int. Conf. Comput. Sci. Inf. Technol. (ICCSIT)*, 2009, pp. 48–52.

[16] J. Liu, N. Kato, J. Ma, and N. Kadowaki, "Device-to-device communication in LTE-advanced networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 1923–1940, 4th Quart., 2015.

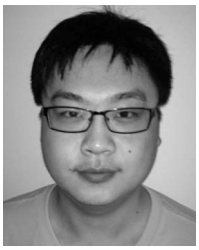
[17] J. Liu, Y. Kawamoto, H. Nishiyama, N. Kato, and N. Kadowaki, "Device-to-device communications achieve efficient load balancing in LTE-advanced networks," *IEEE Wireless Commun.*, vol. 21, no. 2, pp. 57–65, Apr. 2014.

[18] K. Liu *et al.*, "Development of mobile ad-hoc networks over Wi-Fi direct with off-the-shelf Android phones," in *Proc. IEEE Int. Conf. Communications (ICC)*, 2016, pp. 1–6.

[19] S. U. Masruroh, I. Saputra, and Nurhayati, "Performance evaluation of instant messenger in Android operating system and iPhone operating system," in *Proc. IEEE Int. Conf. Cyber IT Service Manag.*, 2016, pp. 1–6.
[20] N. B.-N. I. Minar and M. Tarique, "Bluetooth security threats and solutions: A survey," *Int. J. Distrib. Parallel Syst.*, vol. 3, no. 1, p. 127, 2012.
[21] L. K. Raju and R. Nair, "Secure hotspot a novel approach to secure public Wi-Fi hotspot," in *Proc. IEEE Int. Conf. Control Commun. Comput. India*, 2015, pp. 642–646.
[22] W. Shen, B. Yin, X. Cao, L. X. Cai, and Y. Cheng, "Secure device-to-device communications over WiFi direct," *IEEE Netw.*, vol. 30, no. 5, pp. 4–9, Sep./Oct. 2016.
[23] M. Wang and Z. Yan, "A survey on security in D2D communications," *Mobile Netw. Appl.*, vol. 22, no. 2, pp. 195–208, 2017.
[24] S. Wang *et al.*, "Opportunistic routing in intermittently connected mobile P2P networks," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 9, pp. 369–378, Sep. 2013.
[25] X. Xia, C. Qian, and B. Liu, "Android security overview: A systematic survey," in *Proc. IEEE Int. Conf. Comput. Commun. (ICCC)*, 2016, pp. 2805–2809.

Kecheng Liu (S'15) received the B.E. degree in electrical and computer engineering from the University of Illinois at Urbana–Champaign, Champaign, IL, USA, in 2013, and the M.S. degree in computer engineering from the Illinois Institute of Technology, Chicago, IL, USA, where he is currently pursuing the Ph.D. degree at the Department of Electrical and Computer Engineering.

His current research interests include Internet of Things system security and device-to-device network security.



Wenlong Shen (GS'13–M'16) received the B.E. degree in electrical information engineering from Beihang University, Beijing, China, in 2010, the M.S. degree in telecommunications from the University of Maryland at College Park, College Park, MD, USA, in 2012, and the Ph.D. degree in electrical and computer engineering from the Illinois Institute of Technology, Chicago, IL, USA, in 2018.

His current research interests includes information security, D2D communications, and wireless networking.

Yu Cheng (S'01–M'04–SM'09) received the B.E. and M.E. degrees in electronic engineering from Tsinghua University, Beijing, China, in 1995 and 1998, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2003.

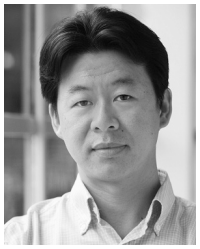
He is currently a Full Professor with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL, USA. His current research interests include wireless network performance analysis, network security, big data, cloud computing, and machine learning.

Dr. Cheng was a recipient of the Best Paper Award at QShine 2007, the IEEE ICC 2011, the Runner-Up Best Paper Award at ACM MobiHoc 2014, the National Science Foundation CAREER Award in 2011, and the IIT Sigma Xi Research Award in the Junior Faculty Division in 2013. He has served as the Symposium Co-Chair for IEEE ICC and IEEE GLOBECOM, and the Technical Program Committee Co-Chair for WASA 2011 and ICNC 2015. He was a founding Vice Chair of the IEEE ComSoc Technical Subcommittee on Green Communications and Computing. He was an IEEE ComSoc Distinguished Lecturer from 2016 to 2017. He is an Associate Editor for the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY.

Lin X. Cai (GS'09–M'10) received the M.A.Sc. and Ph.D. degrees in electrical and computer engineering from the University of Waterloo, Waterloo, ON, Canada, in 2005 and 2010, respectively.

She was a Post-Doctoral Research Fellow with the Electrical Engineering Department, Princeton University, Princeton, NJ, USA, in 2011. She joined the Huawei U.S. Wireless Research and Development Center, as a Senior Engineer in 2012. She has been an Assistant Professor with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL, USA, since 2014. Her current research interests include green communication and networking, broadband multimedia services, and radio resource and mobility management.

Dr. Cai was a recipient of the Post-Doctoral Fellowship Award from the Natural Sciences and Engineering Research Council of Canada in 2010, the Best Paper Award from IEEE Globecom 2011, and the NSF Career Award in 2016. She is an Associate Editor of *IEEE Network Magazine* and the IEEE TRANSACTION ON WIRELESS COMMUNICATIONS and the Co-Chair for IEEE conferences.



Qing Li (M'03) is a Vice President of Engineering with the Network Protection Products Business Unit, Symantec Cooperation, Mountain View, CA, USA. He served as the Chief Scientist with Blue Coat Systems, Sunnyvale, CA, USA, where he is a well-known V1.0 technology and product innovator. He is an industry veteran with over 20 years of experience, with 28 issued patents and many more pending. He is a published author of five first-of-their-kind books, by Springer-Verlag, Morgan Kaufmann, and Wiley. His books have been translated into multiple foreign languages and are serving as reference texts in universities and in industry around the world.

Sheng Zhou (GS'05–M'11) received the B.E. and Ph.D. degrees in electronic engineering from Tsinghua University, Beijing, China, in 2005 and 2011, respectively.

In 2010, he was a Visiting Student with the Wireless System Lab, Department of Electrical Engineering, Stanford University, Stanford, CA, USA, for five months. From 2014 to 2015, he was a Visiting Researcher with Central Research Laboratory, Hitachi Ltd., Tokyo, Japan. He is currently an Associate Professor with the Department of Electronic Engineering, Tsinghua University. His current research interests include cross-layer design for multiple antenna systems, mobile edge computing, and green wireless communications.

Zhisheng Niu (M'98–SM'99–F'12) received the graduation degree from Beijing Jiaotong University, Beijing, China, in 1985, and the M.E. and D.E. degrees from the Toyohashi University of Technology, Toyohashi, Japan, in 1989 and 1992, respectively.

From 1992 to 1994, he was with Fujitsu Laboratories Ltd., Tokyo, Japan. In 1994, he joined Tsinghua University, Beijing, China, where he is currently a Professor with the Department of Electronic Engineering. His current research interests include queueing theory, traffic engineering, mobile Internet, radio resource management of wireless networks, and green communication and networks.

Dr. Niu was a recipient of the Outstanding Young Researcher Award from the Natural Science Foundation of China in 2009 and the Best Paper Award from the IEEE Communication Society Asia–Pacific Board in 2013. He has served as the Chair of the Emerging Technologies Committee from 2014 to 2015, the Director for Conference Publications from 2010 to 2011, and the Director for the Asia–Pacific Board from 2008 to 2009 of the IEEE Communication Society, and is currently serving as the Director for Online Contents from 2018 to 2019 and an Area Editor for the IEEE TRANSACTIONS ON GREEN COMMUNICATIONS AND NETWORKING. He was also selected as a Distinguished Lecturer of the IEEE Communication Society from 2012 to 2015 and IEEE Vehicular Technologies Society from 2014 to 2018. He is a Fellow of the IEICE.