Poster: Insecurities of D2D and a Usable Solution

Jay Prakash
SUTD, Singapore

Andrei Bytes
SUTD, Singapore

Saket Chandra
SUTD, Singapore

Richard Hsu
NSYSU, Taiwan

Jemin Lee
DGIST, South Korea

Tony Q. S. Quek
SUTD, Singapore

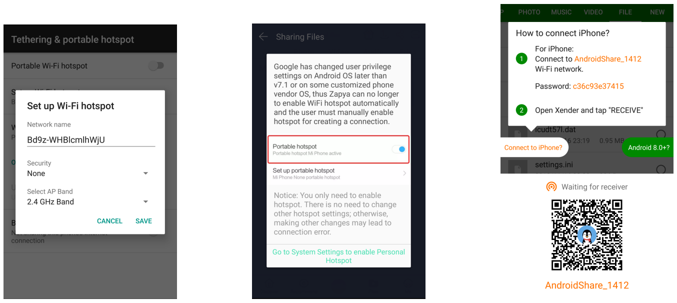
Jianying Zhou
SUTD, Singapore

I. Introduction

The adoption of the fifth generation of cellular mobile communication (5G) is expected to drive cellular networks from centralized to a device-centric infrastructure. D2D is expected to be a dominant mode of communication in 5G, where both the cellular, in-band, and out-band D2D co-exist [1]. D2D facilitates a direct connection between compatible radio-frequency (RF) devices without the need for association with access points (APs) or cellular base stations (BSs). With introduction of Wi-Fi direct, with property of maintaining simultaneous primary Wi-Fi connection, by Wi-Fi Alliance and its integration by Google into Android 4.0 [2], its user base and use cases have increased exponentially over past years [3]. The adoption of Wi-Fi Direct in consumer devices (tablets, smartphones, and smart TVs) has already reached 1.7 billion in 2016, and is predicted to reach 3 billion devices in 2019 [4]. A large portion of D2D connections will have human-in-loop as well. Hence it is imperative to design protocols and applications while keeping track of both usability and security. In this paper we study popular D2D applications developed for smart-phones, identify critical usability and security trade-offs, report vulnerabilities, and propose a framework which helps quantify usability for security researchers and develop new application which is both secure and usable.

II. Current State and Problem Description

To capture current state of D2D ecosystem we studied 6 most popular smartphone applications for D2D sharing: SHAREit (1.5 billion users), Xender(100 million), Xiaomi Mi Drop (> 100 million), Google Files (> 10 million), Zapya (>50 million) and SuperBeam (>10 million). The study involved rigorous security and usability analysis. To our surprise and utmost worry, we found 15 critical vulnerabilities, reported the same and got common vulnerabilities and exposures (CVE) numbers. Google even acknowledged the issue with Google Files, rewarded bounty and took essential measures as well.


Tethering & portable hotspot
Portable Wi-Fi hotspot
CANCEL
SAVE

Set up Wi-Fi hotspot

Network name: Bd9z-WHBlcmlhWjU

Security: None

Select AP Band: 2.4 GHz Band

(a) SHAREit 4.5.84 overwriting the hotspot security to None

Sharing Files
Go to System Settings to enable Personal Hotspot

Google has changed user privilege settings on Android OS later than v7.1 or on some customized phone vendor OS, thus Zapya can no longer to enable WiFi hotspot automatically and the user must manually enable hotspot for creating a connection.

Portable hotspot
Portable hotspot Mi Phone active

Set up portable hotspot
Mi Phone None portable hotspot

Notice: You only need to enable hotspot. There is no need to change other hotspot settings; otherwise, making other changes may lead to connection error.

(b) Zapya 5.7 preventing the user from securing the hotspot

How to connect iPhone?

1 For iPhone:
Connect to AndroidShare_1412
Wi-Fi network.

Password: c36c93e37415

2 Open Xender and tap "RECEIVE"

(c) Xender - QR codes as a side channel for password transmission

Fig. 1. Usability decisions and security misconfiguration

Preference of Usability over Security: A number of key design decisions have been made in the applications, reviewed in this section, to *prioritize* the simplicity of user flow over basic security principles. SHAREit declares the requirement to have bluetooth on *to increase the connection speed*, however, it tends to automatically associate with the hotspot if devices are connected via bluetooth. Xender and Superbeam prioritize generation and reading of QR codes as a primary way to exchange the credentials, needed for sender and receiver to associate (Fig. 1(c)), and encapsulates the credentials into the URI. Surprisingly, our comparative testing has shown that every application in our shortlist, including Google Files 1.0.220185905 for Android implements fall-backs, thus not using Wi-Fi Direct in all times, the applications turn either the sender or receiver into AP, with which the other party is associated programmatically. SHAREit uses UDT protocol and rely solely on link layer security configuration, thus lacking any encryption or integrity protection for transferred files of its users. It turned out that in web sharing mode none of the applications, except Google Files, provide SSL \ TLS or any other modes to protect the confidentiality of the transferred files, exposing the communication to the in-network attacker in clear-text. At times (Android 7.1), ShareIT 4.5.84 , Xender 4.2.2.Prime and Zapya 5.7 (US) set the Android settings dialog with open AP (security: none) and ask for the users action to enable it. Moreover, in the case when the user preconfigures a hotspot with WPA2 *passphras*e, above mentioned applications would override these settings and reset the security mode to None (Fig. 1(a)). Zapya explicitly warns the user to prevent from making any changes to the AP configuration, (Fig. 1(b)). Ignoring this warning and manually setting a hotspot password in the settings dialog resulted in malfunction of ShareIT 4.5.84 and Zapya 5.7, a client is unable to connect to protected APs. Similarly, Xender 4.2.2. Prime connection dialog doesn‘t allow to associate with a protected AP if its password is

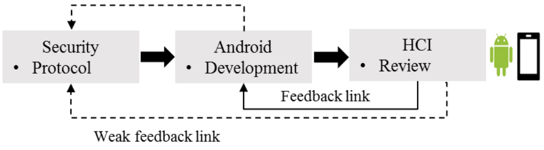
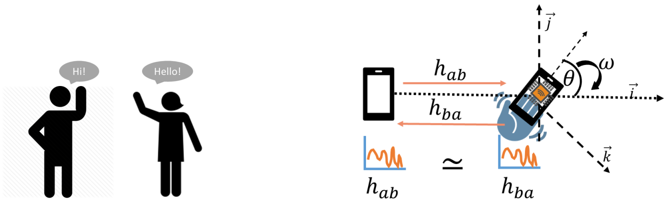


Fig. 2. Standard end-to-end security protocol design



(a)

(b)

Fig. 3. Wave2Share: Analogous to talk initiation in socio-physical world, Hi!

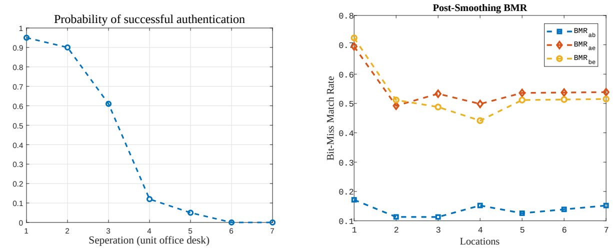
longer than 8 symbols. Xiaomi Mi Drop applies the same workaround for Android 7.1+, however, it bootstraps the AP with a predefined password, which is predictable for the client. Changing this password results in client connection failure, which is analogous to SHAREit and Zapya.

Cause of Trade-off: In order to gain on usability, serious security compromises has been made. There exist a tradeoff between security and usability and one of the main reason for trade-off, apart from improving user experience, is perceived *absence of a source of shared randomness*. D2D connections happen on fly and application designers could not find a dedicated source for mutual authentication and shared secret key which would not involve users participation, like setting up a hotspot, configuring keys and sharing them with other party. We want to highlight that Wi-Fi direct inherently has provisions for authentication and security. But according to current practice, protocol development and deployment is decoupled with user-experience. As shown in Fig.2 there exists no or very weak link between protocol designers and researchers from human-computer-interaction community. As such a dedicated framework, which can be referred to by networking and security scientists, does not exist.

III. Our Solution: Wave2Share

In order to diminish security-usability trade-off, we study basic human behavior and the possibility of its adaptation to security protocols. As shown in Fig.(3(a)), human beings greet each other and wave hand as a gesture. We imbibe the secret generation method in such hand-waving as an attempt to lessen the gap between the *digital* and *physical world* and remove the necessity of entering a secret code or QR scanning. Apart from identification of security issues, main contribution of Wave2Share is to propose a physical layer-powered keyless security paradigm, with implicit security and zero-conf requirement. In doing so we propose:

a) System journey map (SJM): a first of its kind framework where usability can be encapsulated and *quantified* by security researchers themselves which would help them to adapt and analyze usability right in beginning.



(a) Probability of authentication

(b) Secret key mismatch rate

Fig. 4. Authentication and Secret Bit mismatch rate with separation

b) Proximity Assisted Authentication: We propose to exploit physical-cum-social proximity of users (smartphones) to authenticate association with the target device. In doing so, we use wireless channels, from all visible access points. The methodology is based on hypothesis that if RF node pairs are at a close distance ~ $1 - 2m$ they have similar channel observations. The probability pf authentication vanishes sharply with separation Fig. 4(a).

c) Polarization assisted shared secret extraction: Once devices have authenticated each other, they exploit reciprocity in wireless channel between each other, similarity in to-and-fro channel randomness, which is further amplified due to variation in antenna polarization as the user waves his/her hand, to extract shared secret keys as shown in Fig. 3(b). The results are promising since both securely generate similar secret bits with very low bit mismatch rate in presence of an eavesdropper, Fig. 4(b).

The goal of this work is to propose a framework which facilitates designing of secure protocols where user interactions points are cognitively light and demand least effort i.e., security comes from our pre-existing *habits*. As of now it is first of its kind application, which exploits high AP density for proximity authentication and leverages randomness inherent in Wi-Fi direct wireless channels, induced due to variation in antenna polarization at behest of human gesture, for secure D2D communication link on commercial Android smart-phones. We do tests and analysis of security and usability of Wave2Share using SJM under different environments and with user profile.

References

[1] F. Jameel, Z. Hamid, F. Jabeen, S. Zeadally, and M. A. Javed, “A survey of device-to-device communications: Research issues and challenges,” *IEEE Communications Surveys Tutorials*, vol. 20, no. 3, pp. 2133–2168, thirdquarter 2018.

[2] “Android direct.” [Online]. Available: https://developer.android.com/training/connect-devices-wirelessly/wifi-direct

[3] P. Gandotra and R. K. Jha, “Device-to-device communication in cellular networks: A survey,” *Journal of Network and Computer Applications*, vol. 71, pp. 99–117, 2016.

[4] A. Altaweel, R. Stoleru, and G. Gu, “Evildirect: A new wi-fi direct hijacking attack and countermeasures,” in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, July 2017, pp. 1–11.