Demonstration of A Novel Storage Covert Channel on Android Smartwatch Using Status Bar Notifications

Kyle Denney, A. Selcuk Uluagac, Kemal Akkaya, and Nico Saputro
Dept. of Electrical & Computer Engineering
Florida International University
Miami, FL, 33174 USA
kdenney@fiu, auluagac@fiu.edu, kakkaya@fiu.edu, nsapu002@fiu.edu

***Abstract*—Covert channels have been used as a means to circumvent security measures and leak sensitive data undetectable to an onlooker. While these channels have been used in a lot of networks, their applications on mobile devices have just started to be seen. Recently, many covert channels in Android systems have been presented utilizing various available system resources. This demo aims to show how a new storage covert channel can be implemented on wearable devices to leak data from them to outsiders by utilizing the notifications that are normally displayed on the status bar of an Android device. In this demo, we will present the detailed design and implementation of the covert channel for Android based wearable devices. Our demonstration will present the functionality and feasibility of the proposed covert channel on real devices.**

***Keywords*—Android, covert channel, security, mobile, wearables covert channels, Internet-of-Things covert channel**

I. Introduction

In recent years, Internet-of-Things (IoT) devices, have become increasingly popular. It is predicted that by 2020, there will be 50 to 100 billion devices connected to the Internet [6], [5], forming a massive IoT. This emerging IoT technology will drastically change our daily lives and enable smarter cities, health, transportation, and energy [5] for a better and efficient living. Among these devices, a considerable number of them will be the wearable devices that can be carried by individuals such as glasses, watches, sensors, etc. By 2019, it is estimated that 1 in 4 smartphone owners will also be using a wearable device [4]. With new technology comes new security risks, wearables are no exception.

One of these security risks is about covert channels [3], [2], [11]. These are typically used to subvert security measures and steal sensitive information that is undetectable to an onlooker. In a general sense, covert channels utilize normal system resources that are available universally to processes without needing an extra resource for their malicious purposes. For instance, in a *storage covert channel* [11] one malicious process can change or alter a value on the system resource using a

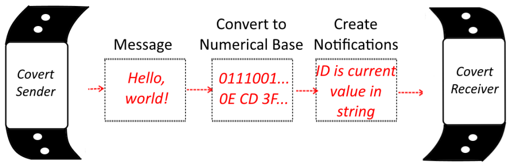

Fig. 1: Design of the covert channel

predetermined codebook. Another malicious process can then read the values that were changed and interpret the coded message that was sent. Two or more processes can use this method to work surreptitiously in tandem to send data to a malicious process (see Fig 1. For example, two applications can work together to steal contact list from a smart phone. The covert sender converts the contacts into string data and transfers the information to the covert receiver. The receiver can then send the data to a malicious server for an attacker to obtain it.

In the Android operating system, many covert channels have been discovered recently [3]. They use resources such as changing volume settings, brightness settings, vibrations [1], or even CPU timings. This demo aims to introduce a new storage covert channel that works by utilizing notifications in the status bar in wearable devices. The covert sender application on a smart phone or device creates a series of notifications that conform to a preset codebook. The covert receiver application then reads the notifications as they appear on the smartwatch paired with the device and interprets the coded message that was sent. In this demo, we introduce the design of this novel notification-based storage channel for Android wearable devices using real smartwatches.

II. Background Information

A. Android Covert Channels

Covert channels can be created on multiple types of devices. Due to the popularity of mobile devices, an increasing number of covert channels are coming out that utilize these mobile devices [1], [3]. In this demo, since we focus on Android devices, we first introduce how covert channels can function on an Android device.

The Android operating system is based on the Linux Operating System (OS) kernel. Therefore, Android follows many aspects of how Linux OS implements its security procedures. Android uses Sandboxing by giving each application a unique user ID. This user ID is then paired with its own set of permissions and system resources. Since each application is a different user, applications are not allowed to share information among themselves.

Some covert channels may attempt to subvert these sandboxing methods. Applications may form either a timing-based or a storage-based covert channel. One application can write information by altering a system resource while the second application can read that information as the resource is changed. Since system resources are universally given to applications, these attacks circumvent the security measures Android employs.

B. *Notifications on Wearables*

The covert channel that is presented in this demo uses the *notification* mechanism available on Android-based wearables. Notifications are messages that are displayed to the user to give information about an application that is running. Examples of notifications would be a new SMS or email message, news, or information about a system update for the device itself.

In order for notifications to be pushed from an Android phone to a wearable (or vice versa), a notification listening service on the wearable needs to be implemented. When the user gives a permission called BIND_NOTIFICATION_LISTENER_SERVICE to an application, the application is allowed to access information about every notification that appears on the status bar. If the user has a wearable device synced with their smart phone, the permission allows for all notifications across both devices to be read.

III. Using Wearable Notifications as a Covert Channel

The covert channel that is implemented in this demo creates notifications using varying notification ID numbers as the coded message. A covert receiving application reads the incoming notifications, detects the ID numbers of each notification, and interprets the covert message (Figure 2). Due to using the notification ID to transmit the covert message, the text of the notification can be disguised to increase the secrecy of the communication.

Functions that are utilized in this covert channel protocol are the *notify(int ID, notification Notification)* and

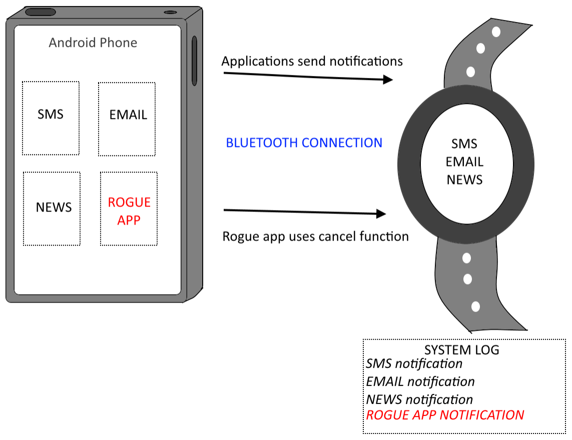

Fig. 2: Overall Sync Architecture between the Phone and Wearable.

cancel(int ID) functions. *Notify* allows the application to call a pre-declared notification and use a specific ID value for the notification. The way the function is structured allows for a notification to be fetched multiple times by using different ID numbers while still using the same notification. The *cancel* function removes a notification with a specified ID from the status bar. Therefore, if the *notify* function is called and is immediately followed by the *cancel* function, the notification is never displayed to the user although it can be seen in the log files as can be seen in Fig. 2. Rogue app notifications will not be displayed to the user but in fact they are in the logs.

When transmitting data, the covet sender application utilizes three separate notifications: (1) a notification to signal the start of transmission, (2) a notification that represents the data, and (3) a notification to signal the end of the transmission. When data is ready to be transmitted, the starting notification is sent out with a predetermined starting ID. Then, the data notification is called with varying ID values that correlate to the data values in the transmission. When the data has been sent, the termination notification is broadcasted and communication between the applications is halted.

In order to function properly, the covert receiving application requires the use of the BIND_NOTIFICATION_LISTENER_SERVICE permission. This permission allows the application to be running constantly and be allowed to read notifications as they are broadcasted on the system. With this, the covert receiver can detect when the sending application transmits a covert code. When the starting ID is transmitted, the receiver starts collecting data values from the sender. When the finishing ID is detected, the data is collected and converted back into the original message. From there, the covert receiver

Fig. 3: Devices used for testbed - left: Samsung Galaxy S5; right: Sony SmartWatch 3

can send the data to an offsite server to be collected by an attacker.

IV. Goals of The Demo

Our demo will prove that a covert channel can be established by simply utilizing notification ID values. Specifically, we will show that:

- A covert channel is established between two applications on a single smart device.
- The covert channel is not limited to one device and that communication is occurring across multiple synced devices. The covert channel will be implemented on an Android-based smart device and a synced Android wearable device. Communication between two applications on the smart device will be shown followed by confirmation that communication is also present on the synced wearable.

V. Setup

A. Equipment to be used

The equipment that will be used in the demo includes: 1 Android smart phone and 1 Android smart watch as seen in Fig. 3

B. Space Needed-Setup Time

The space needed will be approximately a 2 ft. x 3 ft. table top and we will need 15 minutes to set up.

C. Additional Facilities Needed

We will need two connections to a power outlet with ground. No wired or wireless connection to any network on Internet will be needed.

VI. Conclusion and Future Work

As more wearable consumers want notifications to be shared with across all of their devices, abuse of notifications for malicious purposes to establish covert communication channels will always be prevalent. On the Android marketplace currently, there are applications that allow for notifications to be sent across all the users' devices. These applications are given permission to view all notifications on a smart phone and interpret the information from them. If the application is untrustworthy, it can steal data from the user and send it to an offsite malicious receiver. This demo aimed to show how a notification-based storage covert channel can be realized on wearables.

VII. Acknowledgements

This work was supported by the US National Science Foundation under the grant no: REU-CNS-1461119.

References

[1] Ahmed Al-Haiqi, Mahamod Ismail, and Rosdiadee Nordin. A new sensors-based covert channel on android. *The Scientific World*, 2014, 2014.

[2] Moreno Ambrosin, Mauro Conti, Paolo Gasti, and Gene Tsudik. Covert ephemeral communication in named data networking. In *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security*, ASIA CCS '14, 2014.

[3] Swarup Chandra, Zhiqiang Lin, Ashish Kundu, and Latifur Khan. Towards a systematic study of the covert channel attacks in smartphones. *SECURECOMM 2014*, 2014.

[4] Jonah Comstock. http://mobihealthnews.com/37543/pwc-1-in-5-americans-owns-a-wearable-1-in-10-wears-them-daily/. *Mobi Health News*, 2014.

[5] Peter F. Drucker. Internet of Things position paper on standardization for IoT technologies, Jan. 2015.

[6] Dave Evans. The Internet of Things: How the next evolution of the Internet is changing everything, Apr. 2011.

[7] W. Gasior and Li Yang. Exploring covert channel in android platform. In *Cyber Security (CyberSecurity), 2012 International Conference on*, Dec 2012.

[8] Wade C. Gasior and Li Yang. Network covert channels on the android platform. *Cyber Security and Information Intelligence Research 2011*, 2011.

[9] Butler W. Lampson. A note on the confinement problem. *Communications of the ACM*, 16, 1973.

[10] Yu Fan Liping Ji and Chuan Ma. Covert channel for local area network. In *Wireless Communications, Networking and Information Security (WCNIS), 2010 IEEE International Conference on*, pages 316–319, june 2010.

[11] S.V. Radhakrishnan, A.S. Uluagac, and R. Beyah. Realizing an 802.11-based covert timing channel using off-the-shelf wireless cards. In *Global Communications Conference (GLOBECOM), 2013 IEEE*, pages 722–728, Dec 2013.

[12] Baishakhi Ray and Shivakant Mishra. Secure and reliable covert channel. In *Cyber Security and Information Intellidence Research Workshop*, May 2008.

[13] Gustavus J. Simmons. The prisoners problem and the subliminal channels. In David Chaum, editor, *Advances in Cryptology*, pages 51–67. Springer US, 1984.