Pers Ubiquit Comput (2014) 18:1373–1386
DOI 10.1007/s00779-013-0739-y

ORIGINAL ARTICLE

Dangerous Wi-Fi access point: attacks to benign smartphone applications

Min-Woo Park · Young-Hyun Choi · Jung-Ho Eom · Tai-Myoung Chung

Received: 2 July 2013 / Accepted: 26 September 2013 / Published online: 29 October 2013

Abstract Personalization by means of third party application is one of the greatest advantages of smartphones. For example, when a user looks for a path to destination, he can download and install a navigation application with ease from official online market such as Google Play and Appstore. Such applications require an access to the Internet, and most users prefer Wi-Fi networks which are free to use, to mobile networks which cost a fee. For this reason, when they have no access to free Wi-Fi networks, most smartphone users choose to try to use unknown Wi-Fi access points (AP). However, this can be highly dangerous, because such unknown APs are sometimes installed by an adversary with malicious intentions such as stealing information or session hijacking. Today, smartphones contains all kinds of personal information of the users including e-mail address, passwords, schedules, business document, personal photographs, etc., making them an easy target for malicious users. If an adversary takes smartphone, he will get all of information of the users. For this reason, smartphone security has become very important today. In wireless environments, malicious users can easily eavesdrop on and intervene in communication between an end-user and the internet service providers, meaning more vulnerability to man-in-the-middle attacks. In this paper, we try to reveal the risk of using unknown APs by presenting demonstration results. The testbed is composed of two smartphones, two APs, and one server. The compromised AP forwards messages of victim smartphone to the fake server by using domain name system spoofing. Thus, the application that is running on the victim smartphone transfers HTTP request to the fake server. As a result, this application displays the abnormal pop-up advertisement, which contains malicious codes and links. Our demonstration shows that merely connecting to compromise APs can make a malicious behavior even the applications are benign.

Keywords Security of smartphone application · Wireless security · Man-in-the-middle attack · Smartphone and ubiquitous computing

M.-W. Park · Y.-H. Choi
Department of Electrical and Computer Engineering, Sungkyunkwan University, 300 Cheoncheon-dong, Jangan-gu, Suwon-si, Gyeonggi-do, Korea
e-mail: mwpark@imtl.skku.ac.kr

Y.-H. Choi
e-mail: yhchoi@imtl.skku.ac.kr

J.-H. Eom
Military Studies, Taejeon University, 62 Daehakro, Dong-Gu, Daejeon, Korea
e-mail: eomhun@gmail.com

T.-M. Chung (✉)
Department of Software, Sungkyunkwan University, 300 Cheoncheon-dong, Jangan-gu, Suwon-si, Gyeonggi-do, Korea
e-mail: tmchung@ece.skku.ac.kr

1 Introduction

The technology of mobile phones has advanced dramatically over the last decade, in both hardware and software. The performance of its hardware is now almost comparable with portable computers. For example, the Samsung Galaxy S3 manufactured in 2012 contains in it a 1.5 GHz dual-core processor and a 2-GB RAM, along with various high-end sensor gadgets such as GPS, gyroscope, and accelerometer. With these technological advancements in hardware, smartphones are now able to perform numerous intelligent functions like, for example, automatically adjusting the brightness of display according to illumination value. A great technological advancement was also made in software. The main role for this was played by major OS manufacturers such as Google, Apple, and Symbian, which have released the open SDK and created official markets for smartphone applications. With the creation and subsequent growth of such application markets, many developers now spend their time and money to invent new smartphone applications, bringing ever more intelligent applications to smartphone users. Thanks to all these technological advancements in hardware and software, mobile phones have now evolved into smart phones.

However, some types of these intelligent applications require internet connections to identify users and store their information in web storage. Some other types of applications also need internet connections for downloading real-time information like real-time traffic information. In gaining such internet connections, many users avoid using mobile networks because they cost them money, and instead use free Wi-Fi, sometimes even if the access points for Wi-Fi are unknown.

But, it can be very dangerous connecting their smartphones to unknown access points, because wireless communication, which transmits information on air, is more vulnerable to external intervention than wired communication, which transmits information through cables. This signifies malicious users can more easily eavesdrop on conversations or intercept messages by installing rogue access points.

Today, a smartphone has become a necessity for many of us: we need it to wake up in the morning, check schedules or emails, save memos, and communicate with colleagues through social applications. Because of these broad uses for everyday life, many smartphone users knowingly and unknowingly save in their phone much of their personal information such as e-mail passwords, schedules, business documents, and personal photographs in their smartphones, making them an easy target for those with malicious intentions. The security of smartphones is now in more danger than ever before, although most people remain unaware of the danger.

In this paper, we seek to show the risk of using unknown Wi-Fi access points. To this end, we demonstrate the MITM attacks, and we show that the benign application can import and display injected HTML documents. For understanding, we describe briefly security model of the smartphone. However, smartphone OSs are not the same. So we would focus on the Android OS because we believe it is more open and thus more vulnerable. The security model of the Android OS has three security holes in this model; (1) Android OS will be left a big responsibility to ignorant user about security (2) permission-based security model is vulnerable about privilege escalation attacks, and (3) permission-based security model is not able to cover application-level vulnerability.

The rest of this paper is organized as follows. Section 2 describes the role of the smartphone in ubiquitous computing environment. We can easily associate the smart devices like smartphone and tablet PC when we imagine the vision of the ubiquitous computing. Section 3 gives the security model of the Android OS and vulnerability of this. And, Sect. 4 shows results of MITM attack demonstration. Finally, Sect. 5 concludes the work.

2 Smartphone in ubiquitous computing

The smartphones have become very important devices in ubiquitous computing. In this section, we describe definition, core requirements of ubiquitous computing, and roles of the smartphone in ubiquitous computing.

2.1 Definition of ubiquitous computing

The word "ubiquitous" derives from the Latin word ubique, which means "present everywhere at the same time." Ubiquitous computing (UbiComp) indicates the environments where people have access at anytime and anywhere to information and communication technology (ICT) system. In other words, a ubiquitous system allows people to surround themselves with computing devices that understand and support their life cycles. The term ubiquitous computing, first used in 1991 by Mark Weiser in his journal [1], has been redefined by various researchers and institutes [1–5]. Mark Weiser first defined UbiComp as actualization of the "virtual reality" by using invisible computing. Friedemann Mattern [2] now redefines it as comprehensive computerization and interconnection of everyday objects. Yvonne Rogers [3] proposes a new definition of the term: changing roles of the users from calming people to engaging people in UbiComp environment.

The Mark Weiser's vision of UbiComp is composed of three devices, as described in Table 1. The devices are

Table 1 The smart device: tab, pad, and board

Devices	Functionalities
Tab	Tab is a window that is extended from user's computer screen
	It can display what is displayed on user's computer screen
Pad	Pad is interface for transmitting commands to the UbiComp
	It can handle all devices that make up the UbiComp
Board	Board is the biggest device that has a yard-size display
	It is used for the sharing of information between people mainly

classified according to the size of their display. A tab, which is about the size of an ID card, is the smallest device, and a board, which is about a yard size, is the biggest one, with the pad between them. The tab is designed for wearable devices, and this is mainly used for personal functions such as calendar and diary. The pad is a hand-hold device, and it is intended to replace paper. Users can easily read, write, and scrap information using pads. And also the pad is a main interface for handling UbiComp. Boards are used for playing video or sharing information between co-workers.

Nowadays, we can easily associate the smartphone and tablet PC when we imagine a tab and a pad. It means that UbiComp has already been partly realized, and the rest may be fulfilled soon.

2.2 Core requirements of ubiquitous computing

The following are core requirements of UbiComp distinct from distributed computing: context-aware computing, ambient and ubiquitous intelligence, and recording, tracking and monitoring [3, 6, 7]. The most important characteristic of UbiComp is context-aware computing. Service provider expects that UbiComp can provide suitable services to a suitable person at a proper moment without user intervention. For this service, computing ability to understand context information about personal and environmental context is required in UbiComp. In order to collect context information, sensor devices which can collect information and transfer collected information to base station are deployed in UbiComp field. We call this computing environment context-aware computing. Context-aware computing is used to infer situation and decide next operation. This process is important because if the inference result does not match up with the actual user's expectations, UbiComp will lose trust from the user. This is a difficult part of research about context-aware computing.

Second core requirement is related to ambient and ubiquitous intelligence. Sometimes, more accurate interface is necessary in UbiComp. For example, when the user wants to adjust the volume of the audio or change the TV channel, he needs accurate interfaces for communication with the UbiComp system. Generally, speech recognition and gesture recognition technology are often used in this area. However, the error rate is still high, and therefore, technical research is needed to enhance accuracy.

The rest of the requirements are recording, tracking and monitoring. These requirements are adopted to develop human-assistive applications through sensing and alerting [3]. UbiComp has sufficient information for tracking and monitoring human resources because sensor devices of context-aware computing periodically report personal and environmental contexts to it. If UbiComp tracks and monitors the vulnerable people such as the elderly, the physically and mentally disabled, UbiComp can respond to emergency situations. However, it has the following problems. First, it is difficult to record, track, and monitor all of the transactions that occur in UbiComp because of the massive amounts of transactions. UbiComp is composed of a lot of sensor devices that collect context information or wait user input. Thus, the massive amounts of transactions occur in short time. Second, recording, tracking, and monitoring personal information are conflicted with the protection of personal information.

2.3 Smartphone in ubiquitous computing

Over the last decade, a mobile phone has made remarkable advancements in both hardware and software. The mobile phone, also called smartphone, is equipped with a high-speed multi-core processor and enough gigabytes storage incomparable with those of the past feature phone. Furthermore, the smartphone has various built-in sensor gadgets such as GPS, an accelerometer, and a gyroscope. As a result, the smartphone has become so intelligent and more user-friendly as to support our life, just like the vision of UbiComp envisioned. Table 2 shows the specification of Samsung Galaxy S3.

And also, there have been great advancements in software technology. Smartphone OS manufacturers like Google, Apple and Symbian, release the SDK for developing smartphone applications. And also, they create official markets for application deployment. With the growth of markets, many developers are now motivated to invent new smartphone applications, producing a large number of useful applications reactive to context by means of multiple sensors built in the smartphone. According to the Android official blog, the Google Play has reached 25 billion downloads and 675,000 total apps [8].

Evolution of the smartphone has greatly changed our lifestyle. From a morning call service to a remote control vehicle service, the smartphone offers various services to the user [9]. The smartphone already plays the roles similar to the tab and the pad as envisioned by Weiser. People at anytime and everywhere carry their smartphones like wearable devices. And the smartphone performs most of

Table 2 The specification of Samsung Galaxy S3

Segments	Specifications
Processor	1.5 GHz dual-core processor
Memory	32 gigabytes of storage and 2 gigabytes of RAM
Connectivity	Bluetooth, Wi-Fi, NFC, etc.
Built-in sensors	Accelerometer, gyroscope, proximity, compass, and barometer

the personal tasks such as scheduling, checking e-mail, and sharing files. It is similar to the tab in the vision of Weiser. And also, the size of smartphone is similar to that of the pad. Moreover, the user writes notes on and clips information to his smartphone. All these technological advances are realization of Weiser's vision [10].

Furthermore, we expect the smartphone to become the most important equipment in UbiComp. The smartphone can be used for satisfying the core requirements of UbiComp. The smartphone can collect context information and transfer collected information to the base station through wireless communication. And also, the smartphone can be used for human interfaces. Lastly, the smartphone can be utilized as an identification of its owner by using universal subscriber identity module (USIM) information.

The smartphone can play the role of deployed sensor devices in context-aware computing. Context-aware system needs wireless sensor networks for collecting context information. Thus, large numbers of sensor devices are deployed in UbiComp to continuously collect context information such as personal and environmental context and transfer it to the base station. However, it is highly costly to construct these sensor networks. Moreover, these sensor devices have energy limitation. So, sensor devices cannot perform permanently. However, most of these problems can be solved by using the smartphone. People always carry their smartphone with them, which contains various built-in sensors. And all smartphones have wireless network interfaces. Thus, the smartphone can easily transfer messages to the base station. Since the smartphone is fully charged on a daily basis, so users can be free of the fear of energy shortage. As a result, the smartphone can play a role of deployed sensor devices in UbiComp field. Andrew et al. [11] and Tor-Morten et al. [12] show several examples of sensing applications for cognitive phones.

Next topic is the suitability of smartphones for Human–Computer interfaces. The smartphone has various interfaces for interactive functions such as camera, touch panel, gyroscope, and up-down buttons. Various technologies already have been used for interaction between human and devices in smartphone. For examples, the smartphone can adjust the brightness of the screen automatically by recognizing the user's eye, and the scroll of web browser is controlled by just tilting the device. Rafael et al. [13] and George et al. [14] suggest a possibility of smartphone usable as an input device in UbiComp. Following them, we expect interaction between humans and UbiComp using the smartphone is possible.

The last core requirements of UbiComp are recording, tracking and monitoring the people. It can reduce the cost spent identifying and tracking the user by using the smartphone. Generally, the smartphone has a USIM card with unique serial numbers, which is issued by mobile network providers for identification of the owner. Thus, if UbiComp can read USIM information, it will be able to easily track and monitor human resources.

Figure 1 illustrates the abstract roles of the smartphone and its interactions with UbiComp environment. The smartphone has resources such as privacy information (e.g., schedule, contact, etc.), built-in sensor devices (e.g., GPS module, gyroscope, accelerometer, etc.), and applications for UbiComp that are optionally installed with the permission of the user. The solid line points to interaction between the human and the computer and the dotted line machine-to-machine interaction. The Context-aware Computing component demands context information of personal and environmental context. So, this component communicates with optional applications to receive personal context. Every component of UbiComp interacts with built-in sensors of the smartphone to collect context information. The Context-aware Computing component and Ambient and Ubiquitous computing component have direction access to sensor devices. These components receive raw data from the smartphone and process them according to their function. On the other hand, the Recording, Tracking and Monitoring component communicates with optional applications because this component requires refined data.

3 Threats of the compromised access points

In this section, we describe the security model of the smartphone through the Android platform and critical security threats posed by the MITM attack, which can occur from compromised AP connections.

3.1 Security model of smartphone

The smartphone has a dual- or quad-core processor and a gigabyte memory and storage. Whenever and wherever users desire, they can gain connections to the Internet using mobile networks or Wi-Fi networks. And, it can access user information stored in the device or on the web. Although functional aspects of the smartphone have grown significantly, the security technology of the smartphone still falls short of expectations of many. Smartphone OSs are slightly different from each other, and we will focus on the Android OS, because it is more open and thus more vulnerable to external invasion. In this subsection, we describe security flaws of the Android OS.

Basically, all applications run within their own sandbox, and no application can escape this sandbox. However, these restrictions are so strong as to cut off most of the functionality of smartphone application. So, the Android platform allows use of the API depending on application

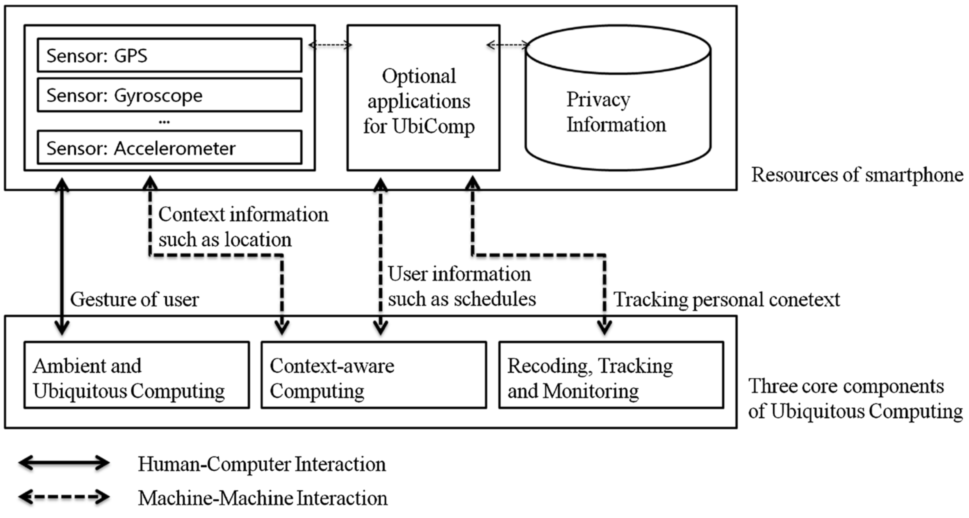
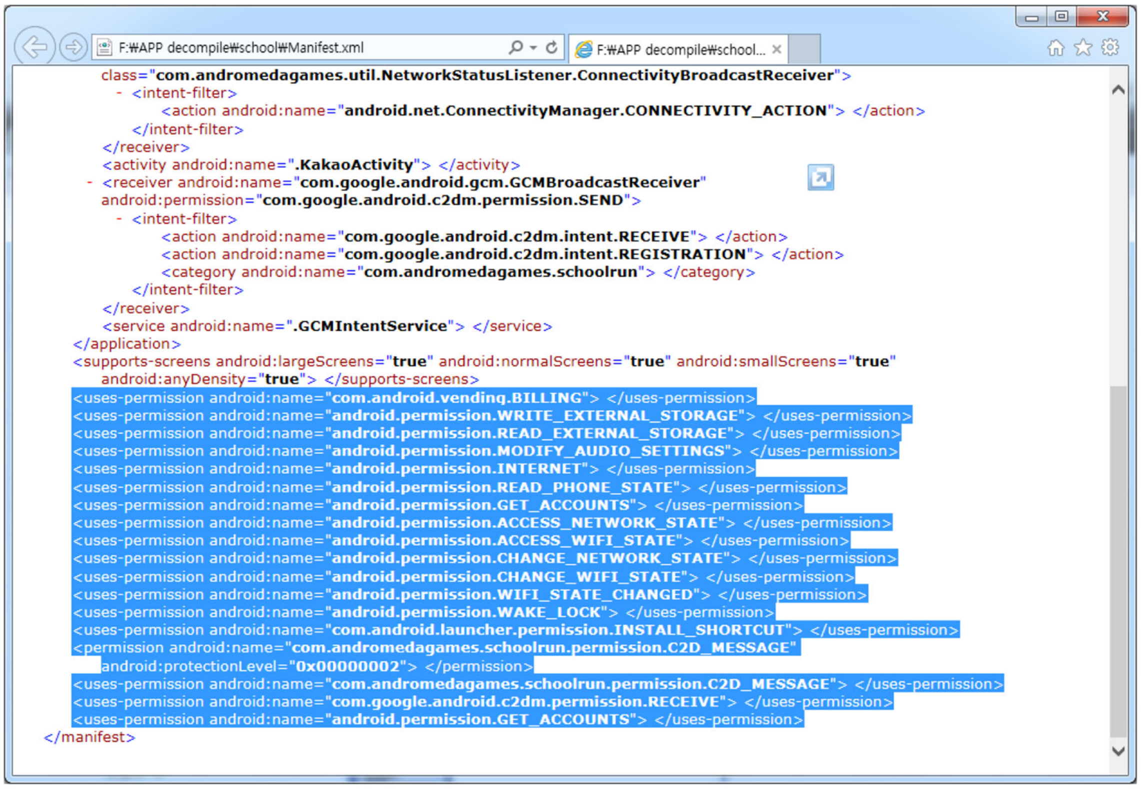

Sensor: GPS
Sensor: Gyroscope
...
Sensor: Accelerometer
Optional applications for UbiComp
Privacy Information
Resources of smartphone
Context information such as location
Gesture of user
User information such as schedules
Tracking personal conetext
Ambient and Ubiquitous Computing
Context-aware Computing
Recoding, Tracking and Monitoring
Three core components of Ubiquitous Computing
Human-Computer Interaction
Machine-Machine Interaction

Fig. 1 The abstract roles of Smartphone and interactions with UbiComp environments


F:₩APP decompile₩school₩Manifest.xml
F:₩APP decompile₩school...

```
    class="com.andromedagames.util.NetworkStatusListener.ConnectivityBroadcastReceiver">
     - <intent-filter>
          <action android:name="android.net.ConnectivityManager.CONNECTIVITY_ACTION"> </action>
       </intent-filter>
    </receiver>
    <activity android:name=".KakaoActivity"> </activity>
  - <receiver android:name="com.google.android.gcm.GCMBroadcastReceiver"
    android:permission="com.google.android.c2dm.permission.SEND">
     - <intent-filter>
          <action android:name="com.google.android.c2dm.intent.RECEIVE"> </action>
          <action android:name="com.google.android.c2dm.intent.REGISTRATION"> </action>
          <category android:name="com.andromedagames.schoolrun"> </category>
       </intent-filter>
    </receiver>
    <service android:name=".GCMIntentService"> </service>
 </application>
 <supports-screens android:largeScreens="true" android:normalScreens="true" android:smallScreens="true"
    android:anyDensity="true"> </supports-screens>
 <uses-permission android:name="com.android.vending.BILLING"> </uses-permission>
 <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"> </uses-permission>
 <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"> </uses-permission>
 <uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS"> </uses-permission>
 <uses-permission android:name="android.permission.INTERNET"> </uses-permission>
 <uses-permission android:name="android.permission.READ_PHONE_STATE"> </uses-permission>
 <uses-permission android:name="android.permission.GET_ACCOUNTS"> </uses-permission>
 <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"> </uses-permission>
 <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"> </uses-permission>
 <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"> </uses-permission>
 <uses-permission android:name="android.permission.CHANGE_WIFI_STATE"> </uses-permission>
 <uses-permission android:name="android.permission.WIFI_STATE_CHANGED"> </uses-permission>
 <uses-permission android:name="android.permission.WAKE_LOCK"> </uses-permission>
 <uses-permission android:name="com.android.launcher.permission.INSTALL_SHORTCUT"> </uses-permission>
 <permission android:name="com.andromedagames.schoolrun.permission.C2D_MESSAGE"
    android:protectionLevel="0x00000002"> </permission>
 <uses-permission android:name="com.andromedagames.schoolrun.permission.C2D_MESSAGE"> </uses-permission>
 <uses-permission android:name="com.google.android.c2dm.permission.RECEIVE"> </uses-permission>
 <uses-permission android:name="android.permission.GET_ACCOUNTS"> </uses-permission>
</manifest>
```

Fig. 2 An example code of permissions that is stored in AndroidManifest.xml file

permissions as approved by the user [15]. Every Android application has permission information that is approved during the install time in its own AndroidManifest.xml file. This permission never changes until the application is re-installed. Figure 2 shows an example code of the Adn-roidManifest.xml file of the Test Application 1 [34]. The permissions are defined separately for each API that has a risk of being exploited. The permission "INTERNET," that is in the Fig. 2, is necessary for connection with Internet. Android OS verifies permission just when the user application calls the API that has a risk of being exploited. In other words, if some application does not call API, that is, related socket, the Android OS will never check the permission "INTERNET."

The permission-based security model of Android OS has the following security holes. First, Android OS will be left a big responsibility to ignorant user about security [16–19]. Most users do not understand about the risk of approving the permission to applications. Furthermore, the user has only the two choices giving an approval or not. As a result, the user thinks less of permission authorizing process because of this permission policy. Second, the permission-based security model is vulnerable to privilege escalation attacks [20, 21]. Multiple applications share the role for achieving their purpose. For an example, malicious application A has permission for accessing to sensitive internal data such as the contact, but it does not have permission for sending message through the Internet. Malicious application B does not have permission for accessing to sensitive internal data, but it has permission about the Internet. In this case, malicious application A is to transmit the contact to malicious application B by using internal communication path and malicious application B flows out the contact through the Internet [22]. Last, the permission base security model is only able to cover low-level behaviors that are related to API call. If android application has application-level vulnerability, the Android OS will not be able to protect itself. We focus on this security hole. In the next section, we demonstrate the MITM attack by using this application-level vulnerability.

3.2 Threats of the compromised access points

According to growth of wireless networks, wireless network interface becomes the most basic parts of a portable computing device. Some research predicts that wireless communications will exceed wired communications by 2015. Like this, wireless communication technology has become the most important communication means for connecting the smart device. The growth of wireless communication contributes to realize UbiComp and popularizes smartphone. However, security threats exist in the hidden side of the rapidly growth of wireless communication.

We can easily see that wireless APs are installed in a narrow area more than needs. Figure 3 shows the map that presents the density of APs in Chicago. We obtain this map from wireless geographic logging engine (WIGLE) project which is a dataset for collecting the wireless hotspots around the world [23, 24]. Wireless APs are distributed in Chicago more difficult to read the map. According to WIGLE project, about 5 million APs exist in California that is a region where the AP is installed most in the United States. Density of the wireless AP is very high considering that the each AP can support a range of up to 150 feet indoors and 300 feet outdoors. Table 3 shows regions and the number of wireless APs. We are surrounded by many wireless APs. It is look like a spider web that is configured in a wireless network. In fact, all of the AP that is searched by our devices is not a benign. When you indiscriminately try to connect to unknown AP, you and your device will be in danger.

In the wired network, the MITM attacks are very difficult attack technique [25]. It is impossible that an adversary

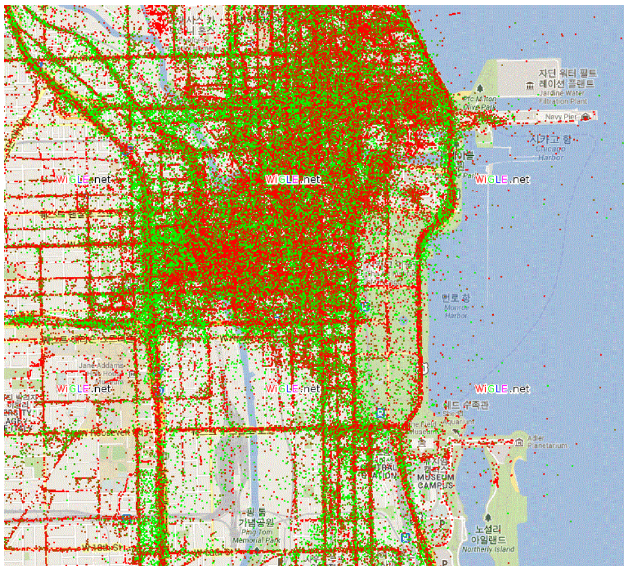


Fig. 3 The map of wireless APs in Chicago

physically break into an end-user and the ISP. Thus, an adversary uses domain name system (DNS) cache poisoning for changing the direction of traffic flows [26] in Fig. 4a. However, in the wireless network, an adversary can easily break into an end-user and the ISP [27–31] in Fig. 4b because the messages are transmitted on air in wireless network. Thus, wireless network is more vulnerable than wired networks.

Generally, the smartphone user wants his smartphone is always connected to the Internet because applications that are installed in his smartphone usually requires the Internet connection for uploading or downloading the real-time information. Thus, the smartphone user often searches open wireless networks. Therefore, if an adversary installs open AP, he can easily connect to victims. The way to install the AP for the MITM attack can be divided into two major types. First, an adversary installs the compromised AP on the public places such as airport, bank, and coffee shop [31]. An adversary can easily catch victims in these places, because the probability of using the smartphone is increased when the people stay a long time in one place. Second method is use rogue AP [27, 28]. Rogue AP is installed outside range of benign AP and masquerade as this benign AP. An end-user is easily cheated because rogue AP use the SSID of the benign AP.

In particular, the smartphone users are required more attention about this unknown APs because the smartphone has became the critical point of user's information security. Generally, all information is included in his smartphone from privacy photographs to business documents. Thus, if the smartphone is compromised by an adversary, the user will suffer socially or financially irreparable damage.

Table 3 Number of wireless AP that is located in United State in Sep 2013

Region	Total	California	Texas	Ohio	New York
Count	40,919,320	5,255,380	3,941,742	2,178,988	2,076,812

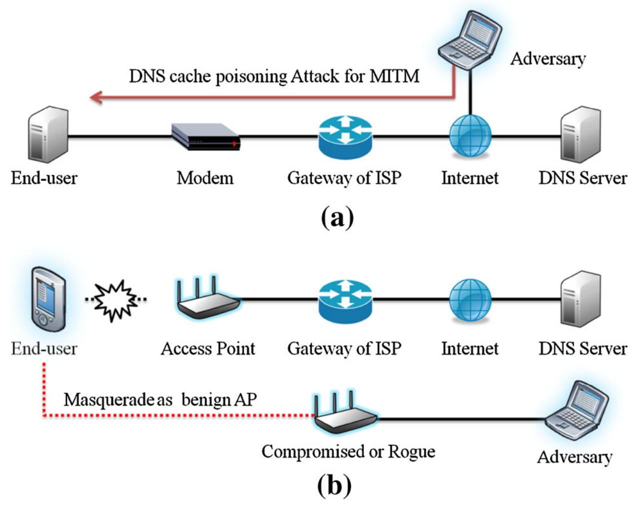


Fig. 4 MITM attack patterns in (**a**) wired and (**b**) wireless communication environment

4 Demonstrations

To indicate the risk of unknown AP, we demonstrate the MITM attack by using compromised AP. We show that an adversary can easily intercept your message and inject modified message into communication between your handset and the service provider. In this section, we describe our demonstration environments and progresses. After then, we explain the results of our penetration test.

4.1 Testbed for the MITM attack

We use five devices for demonstration; two Samsung Galaxy S3s are Android handsets for running applications; a laptop serves as the compromised AP; an Iptimes N40006R is benign wireless AP; and a server for MITM attack. The basic architecture of our testbed is shown in Fig. 5. Two Android handsets connect to each AP through Wireless Local Area Network based on IEEE 802.11. The Android handset 1 is connected to the benign AP, and the other handset is connected to compromise AP. Each AP and the spoofing server which serves as the DNS spoofing server and web proxy server connect to the Internet through same gateway. We set DNS configuration of compromised AP to the spoofing server for DNS spoofing. An adversary is able to catch every packets pass through these compromised AP and divert some packets by using DNS spoofing.

Table 4 shows the tools and software used in our demonstrations. We use top five applications that are registered in “Top New Free Games” of Google Play. These applications import pop-up advertisements of event notification and commercial advertisement from their web servers. Wireshark and Connectify Hotspot are installed on the laptop. Wireshark is used to analyze packets to find vulnerability of communication process. Connectify Hotspot is used to set up the laptop as Wi-Fi AP. Apache2 and Bind is installed on the desktop. Apache2 is used to reply HTTP requests, and Bind is used to deceive Android handset 2.

4.2 Preliminaries

We obtained abstract operations of android applications by analyzing the traffic of applications, as following Fig. 6. Generally, applications communicate with more than one server. First one is a data server. The data server checks application suitability such as user authentication, application version and integrity, and so on. If application fails to test the suitability or access to the date server, then this application is immediately terminated. The second server is an advertisement server, and this server is an optional object. The advertisement server provides html files and image files of event notification and commercial advertisement via HTTP. The connection of the advertisement server does not affect launch of application differently from the connection of the data server. We masquerade as advertisement server for the MITM attack.

Figure 7 shows the MITM attack progress, which consists of passive attack phase and active attack phase.

For the purpose of the passive attack phase, an adversary confirms the existence of the advertisement server and understands communication process between the application and the advertisement server. An adversary monitors the DNS query and response and intercepts the packets between the application and the advertisement server by using the compromised AP. He can easily figure out the IP address and domain name of the advertisement server by using the extracted html documentations and image files through Wireshark. Figure 8 shows an example of the passive attack phase. After receiving a DNS response, the Android handset 2 immediately requests the HTML document to androweb.cafe24.com. This HTML document contains the URL on the pop-up advertisement.

In the active attack phase, an adversary puts the modified HTML documents on the specific path which came

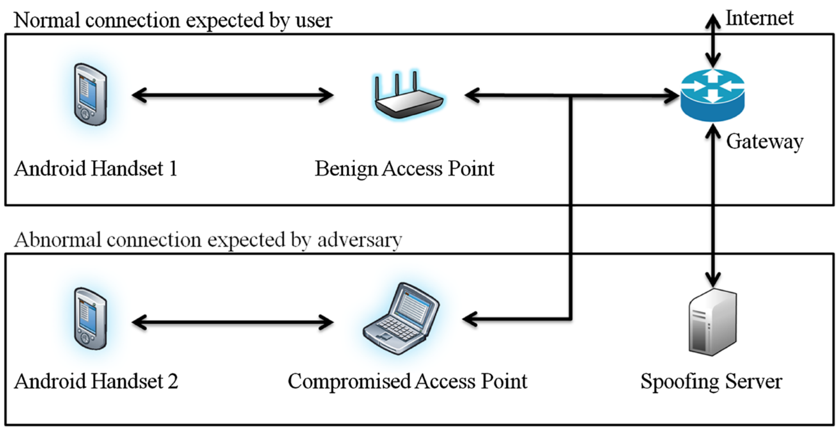


Fig. 5 Architecture of our testbed

Table 4 The tools and software used in our demonstrations

Device	Installed software	Installation purpose
Android handset 1 and Android handset 2	Test Application 1 [35]	Attack demonstration
	Test Application 2 [36]	
	Test Application 3 [37]	
	Test Application 4 [38]	
	Test Application 5 [39]	
Compromised AP	Wireshark	Packets analysis
	Connectify Hotspot	Set up to wireless access point
Spoofing server	Apache2	HTTP proxy server
	Bind	DNS spoofing attack

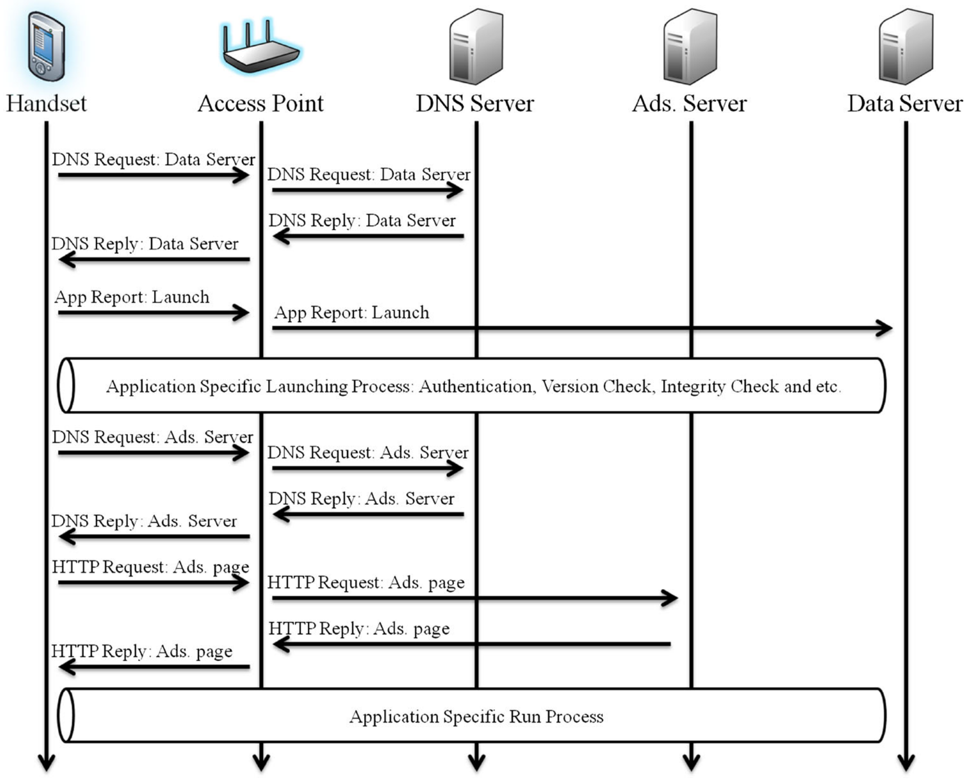
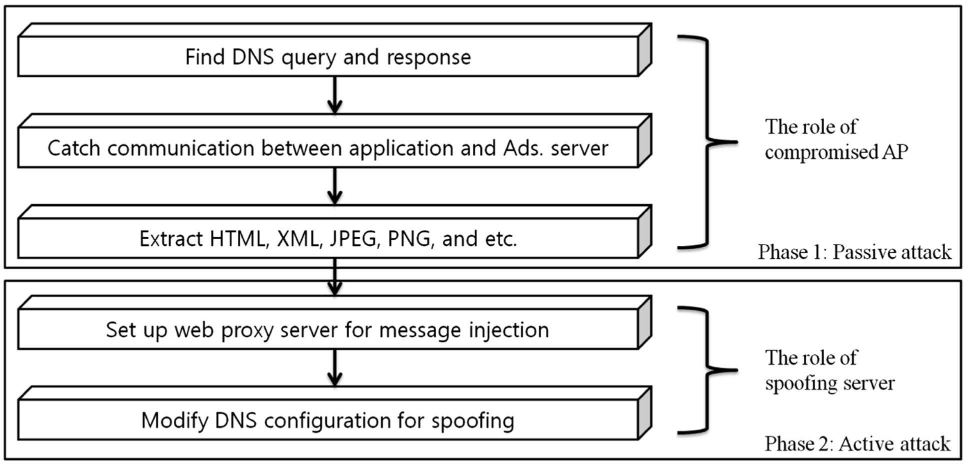
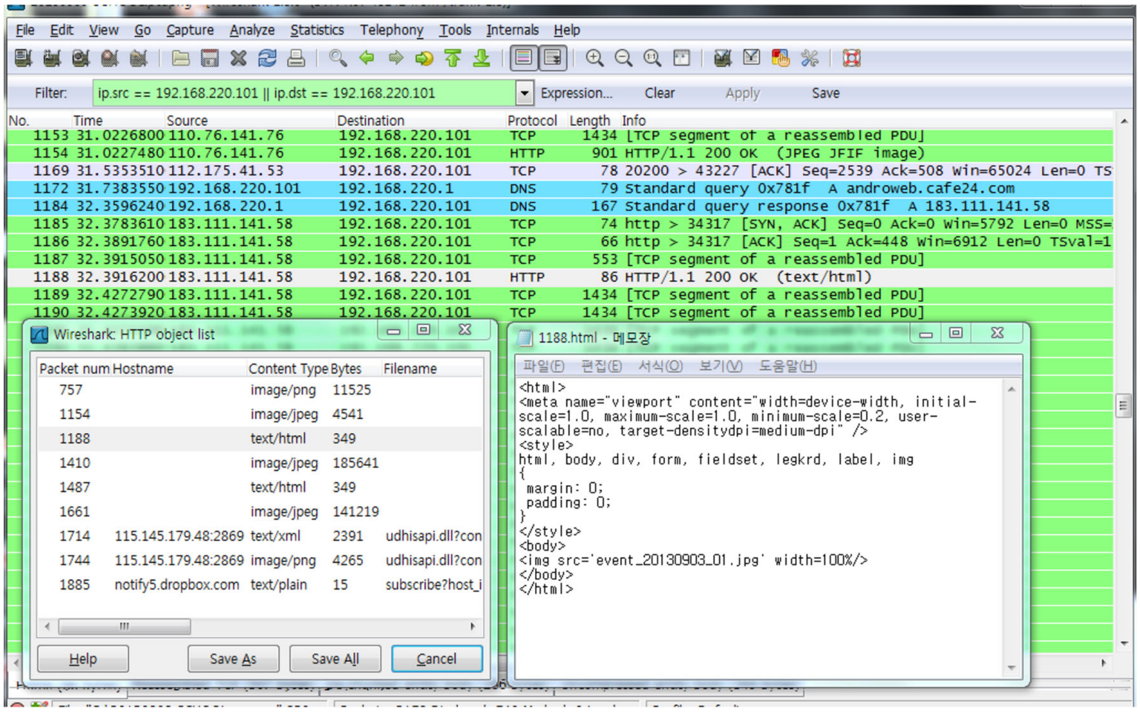


Fig. 6 Abstract operation process of the application which imports pop-up advertisement

Fig. 7 Man-in-the-Middle attack progress


Find DNS query and response
Catch communication between application and Ads. server
Extract HTML, XML, JPEG, PNG, and etc.
The role of compromised AP
Phase 1: Passive attack
Set up web proxy server for message injection
Modify DNS configuration for spoofing
The role of spoofing server
Phase 2: Active attack


File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Expression... Clear Apply Save
Wireshark: HTTP object list
Help Save As Save All Cancel
1188.html - 메모장
파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

```
Filter: ip.src == 192.168.220.101 || ip.dst == 192.168.220.101

No.   Time       Source          Destination      Protocol Length Info
 1153 31.0226800 110.76.141.76   192.168.220.101  TCP      1434 [TCP segment of a reassembled PDU]
 1154 31.0227480 110.76.141.76   192.168.220.101  HTTP      901 HTTP/1.1 200 OK  (JPEG JFIF image)
 1169 31.5353510 112.175.41.53   192.168.220.101  TCP        78 20200 > 43227 [ACK] Seq=2539 Ack=508 Win=65024 Len=0 TS
 1172 31.7383550 192.168.220.101 192.168.220.1    DNS        79 Standard query 0x781f  A androweb.cafe24.com
 1184 32.3596240 192.168.220.1   192.168.220.101  DNS       167 Standard query response 0x781f  A 183.111.141.58
 1185 32.3783610 183.111.141.58  192.168.220.101  TCP        74 http > 34317 [SYN, ACK] Seq=0 Ack=0 Win=5792 Len=0 MSS=
 1186 32.3891760 183.111.141.58  192.168.220.101  TCP        66 http > 34317 [ACK] Seq=1 Ack=448 Win=6912 Len=0 TSval=1
 1187 32.3915050 183.111.141.58  192.168.220.101  TCP       553 [TCP segment of a reassembled PDU]
 1188 32.3916200 183.111.141.58  192.168.220.101  HTTP       86 HTTP/1.1 200 OK  (text/html)
 1189 32.4272790 183.111.141.58  192.168.220.101  TCP      1434 [TCP segment of a reassembled PDU]
 1190 32.4273920 183.111.141.58  192.168.220.101  TCP      1434 [TCP segment of a reassembled PDU]
```

Packet num	Hostname	Content Type	Bytes	Filename
757		image/png	11525	
1154		image/jpeg	4541	
1188		text/html	349	
1410		image/jpeg	185641	
1487		text/html	349	
1661		image/jpeg	141219	
1714	115.145.179.48:2869	text/xml	2391	udhisapi.dll?con
1744	115.145.179.48:2869	image/png	4265	udhisapi.dll?con
1885	notify5.dropbox.com	text/plain	15	subscribe?host_i

```
<html>
<meta name="viewport" content="width=device-width, initial-
scale=1.0, maximum-scale=1.0, minimum-scale=0.2, user-
scalable=no, target-densitydpi=medium-dpi" />
<style>
html, body, div, form, fieldset, legkrd, label, img
{
 margin: 0;
 padding: 0;
}
</style>
<body>
<img src='event_20130903_01.jpg' width=100%/>
</body>
</html>
```

Fig. 8 An example of passive attack phase

from the previous phase. Then he sets up the DNS configuration to divert the HTTP request messages to spoofing server. The code in the Fig. 9 is an example of inserted code in named.conf, that is, configuration file of named which is a DNS server, part of the BIND9 distribution.

4.3 Man-in-the-middle attack progress

Figure 10 shows the MITM attack process. When target applications are launched on the Android handset 2, it checks status of connection to the Internet and tries to request IP address of the data server. DNS query of the Android handset 2 is delivered to the spoofing server passing through the compromised AP. The spoofing server returns the correct IP address of data server for normally launching the target application. Next, the Android handset 2 communicates with the data server for application-specific launching process. Generally, these communications are protected by SSL. If the target application is launched successfully, it tries to request IP address of the advertisement server. In this case, however, the spoofing server returns IP address of itself in order to inject modified messages. As a result, the target application requests HTML documents and image files to the spoofing server and exposes incorrect advertisements that are modified by an adversary.

```
zone "androweb.cafe24.com"{
    type master;
    file "/var/cache/bind/androweb.cafe24.com.zone";
};
```

Fig. 9 An example of inserted code in *named.conf*

4.4 The results of man-in-the-middle attack

As mentioned above, we target top five applications that are registered in "Top New Free Games" of Google Play.

Figure 11 presents screenshot of the target application's pop-up advertisement; the (a) image is a view of normal case, the (b) image is a view of abnormal case that the target application imports modified HTML document, and the (c) image is a screen when the link contained in the modified HTML document has been executed of target application. In many cases, Android applications are used as simple objects such as WebView for pop-up advertisement. As a result, an adversary can easily insert link of the other URLs into modified HTML document. Figure 12 is a part of the code of original HTML document of target application and link-embedded HTML document that is made by us.

Figure 13 shows the results of our MITM attack demonstration. We have inserted "Modified advertisement" to all the images and added the link of the injected HTML document to the original HTML document. We have succeeded in exposing modified advertisement page to user through the all benign applications. And also, the link that is injected by us is working properly in the all test applications.

Figure 14 shows the partial code for generating pop-up advertisement of the Test application 1. WebView is Android API for simple display online content within applications [40]. By using loadUrl method of WebView class, developer can easily handle online content. However, these objects, such as WebView, are in danger of being misused as a result of our demonstration. If developer can to block that execution of embedded link, it is possible to significantly reduce the threat of these attacks.

A spear phishing is more effective because phishing messages are customized for victims [34]. Customized message, which contains trustworthy information such as victim's nickname, is easier to be believed. Thus, the threat of the MITM attack is more critical when it is combined with the social engineering for customizing injected advertisement. Who do not click it when the phrase "Only

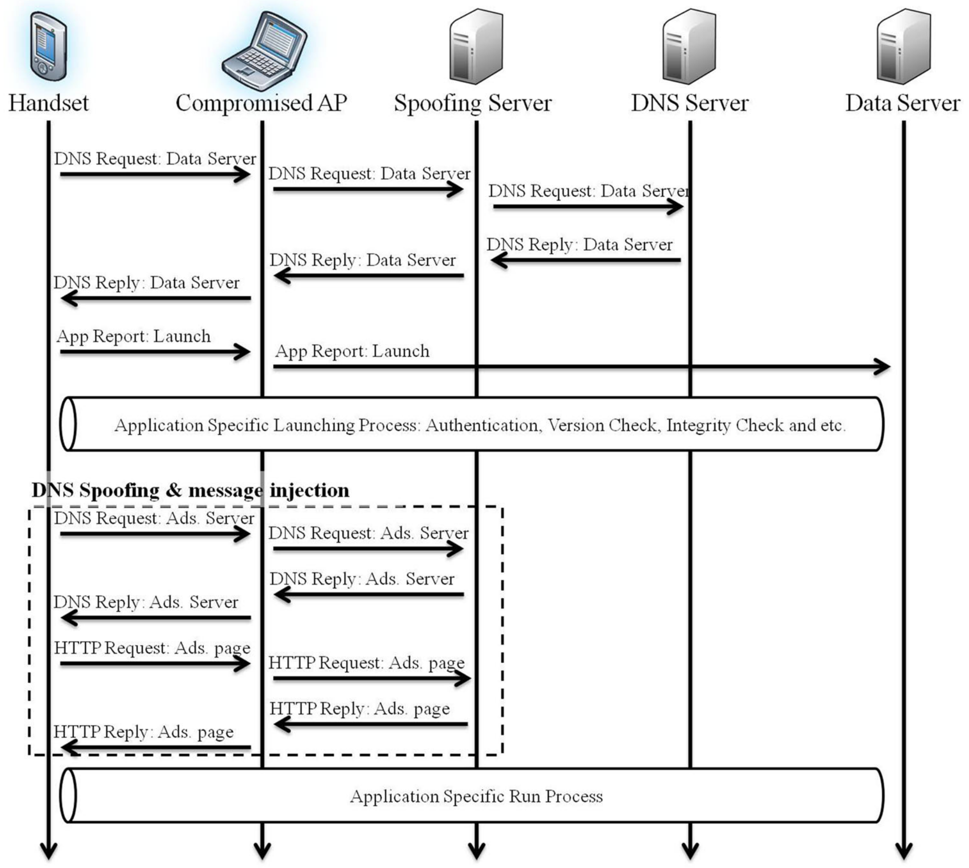
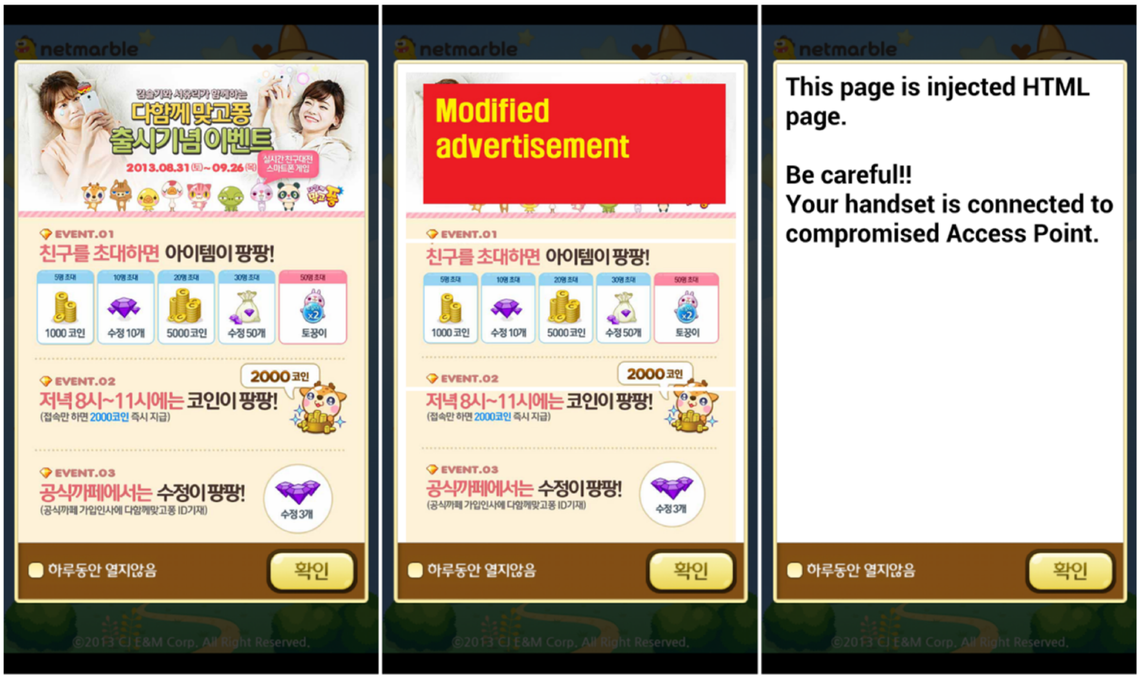


Fig. 10 An example of the MITM attack process


netmarble
EVENT.01
친구를 초대하면 아이템이 팡팡!
EVENT.02
저녁 8시~11시에는 코인이 팡팡!
2000코인
EVENT.03
공식카페에서는 수정이 팡팡!
하루동안 열지않음
확인
©2013 CJ E&M Corp. All Right Reserved.

Modified advertisement

This page is injected HTML page.

Be careful!!
Your handset is connected to compromised Access Point.

(**a**) Normal pop-up Ads. (**b**) Modified pop-up Ads. (**c**) Linked html page

Fig. 11 Captured images of the target application's pop-up advertisement

```
                  •••
<body>
<div class="wrap" data-role="page">
<!-- content -->
<section id="content">
   <div class="evt_wrap">
      <img src="/img/mp1.jpg" width="100%" alt="">
      <img src="/img/mp2.jpg" width="100%" alt="">
      <img src="/img/mp3.jpg" width="100%" alt="">
   </div>
</section>
<!-- //content -->
</div>
</body>
                  •••
```

(**a**)

```
                  •••
<body>
<div class="wrap" data-role="page">
<!-- content -->
<section id="content">
   <div class="evt_wrap">
      <a href="injected.html">
      <img src="/img/mp1.jpg" width="100%" alt="">
      <img src="/img/mp2.jpg" width="100%" alt="">
      <img src="/img/mp3.jpg" width="100%" alt="">
   </a></div>
</section>
<!-- //content -->
</div>
</body>
                  •••
```

(**b**)

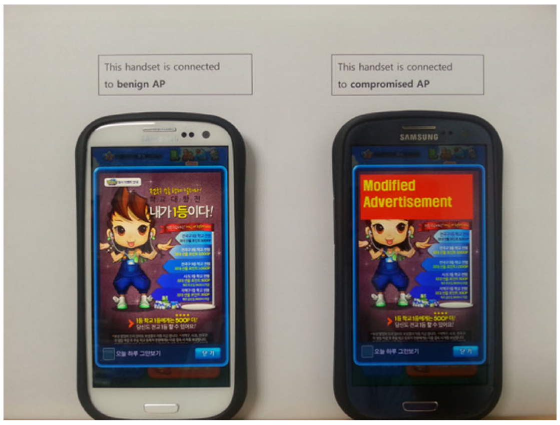
Fig. 12 An example code of the (**a**) original HTML document and (**b**) link-embedded HTML document

one chance! Click and Receive Gift" is inserted in the pop-up advertisement of well-known application?

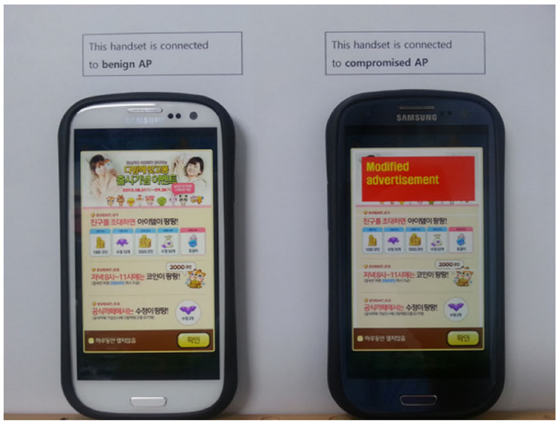
It is necessary to mitigate the MITM attack as follows: (1) the smartphone user avoids connecting to unknown AP, and (2) pays attention to pop-up advertisement even if it is pop-up message of well-known application. (3) The application developer avoids using vulnerable API, and (4) must use mutual authentication process and secure protocol such as SSL [33] when an application communicates with external devices.

5 Conclusions

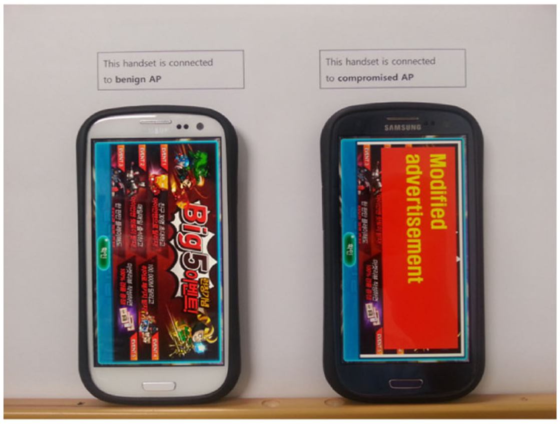
We describe the roles and potentiality of smartphone in the UbiComp environments. The smartphone has done a remarkable development enough to satisfy core requirements of UbiComp: *context-aware computing*, *ambient and ubiquitous intelligence*, and *recording, tracking and monitoring environments*. However, the growth of the smartphones is sufficient to attract the attention of adversaries. Moreover, the security model of the Android


This handset is connected to benign AP
This handset is connected to compromised AP
Modified Advertisement

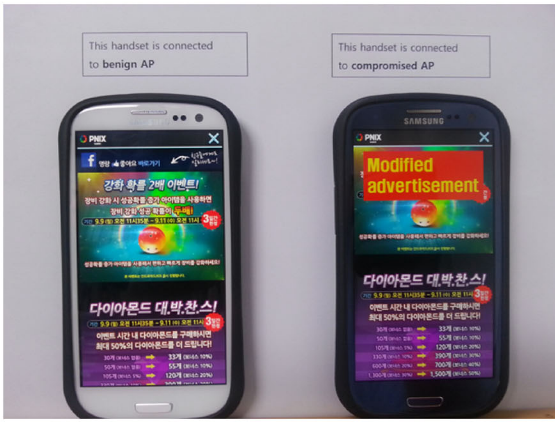
(a) a demonstration result of Test Application 1


This handset is connected to benign AP
This handset is connected to compromised AP
Modified advertisement

(b) a demonstration results of Test Application 2


This handset is connected to benign AP
This handset is connected to compromised AP
Modified advertisement

(c) a demonstration result of Test Application 3


This handset is connected to benign AP
This handset is connected to compromised AP
Modified advertisement

(d) a demonstration result of Test Application 4

Fig. 13 The results of demonstration of the test application 1–4

```
private WebView mWebView;
                                        •••
protected void onCreate(Bundle paramBundle)
{
   super.onCreate(paramBundle);
                                        •••
   localWindow.setAttributes(localLayoutParams);
   localWindow.setBackgroundDrawable(new ColorDrawable(0));
   this.mWebView.getSettings().setJavaScriptEnabled(true);
   this.mWebView.setWebViewClient(this.mMyWebClient);
   this.mWebView.setBackgroundColor(-16777216);
   RelativeLayout.LayoutParams localLayoutParams1 = new RelativeLayout.LayoutParams(-2, -2);
   localLayoutParams1.addRule(10, -1);
   localLayoutParams1.addRule(14, -1);
   localLayoutParams1.setMargins((int)(32.0F * f1), (int)(33.0F * f2), (int)(32.0F * f1), (int)(115.0F * f2));
   this.mWebView.setLayoutParams(localLayoutParams1);
   this.mWebView.loadUrl(mUrl);}
                                        •••
}
```

Fig. 14 The partial code for generating pop-up advertisement by using WebView class

platform has security vulnerabilities such as the following: (1) Android OS will be left a big responsibility to ignorant user about security, (2) permission-based security model is vulnerable about privilege escalation attacks, and (3) permission-based security model is not able to cover application-level vulnerability. In this paper, we reveal the risk of the using unknown APs by using demonstration. The testbed is composed to five devices: two android handsets, one laptop, one desktop, and one wireless AP. The android handsets are used for running application. The laptop plays the role of compromised AP; we change its DNS information. The desktop is used for DNS spoofing and web server. We can intercept and inject packets of passes through the laptop. We divert some packets to the desktop by using DNS spoofing. As a result, test applications that are launched on the handset 2 display abnormal advertisements. We shows that benign application, which is running on uncompromised devices, can be exploited just connecting to the compromised AP. To mitigate this MITM attack, developer must use mutual authentication process when an application communicates with external devices.

In future work, we will continue to research about the MITM attack for attack-protected sessions such as SSL/TLS. Many user applications depute security function to SSL APIs. However, vulnerabilities about these applications have been reported in research of Georgiev et al. [32]. For the development of smartphone security, we continuously study for finding out the vulnerability of smartphone platform and resolving these threats.

References

1. Weiser M (1991) The computer for the 21st century. Sci Am 265(3):94–104
2. Mattern F (2001) The vision and technical foundations of ubiquitous computing. Upgrade 2(5):3–6
3. Rogers Y (2005) Moving on from Weiser's vision of calm computing: engaging UbiComp experiences. In: proceedings of UbiComp 2005. Springer, NY, pp 404–421
4. Leem CS, Jeon NJ, Choi JH, Shin HG (2005) A business model (BM) development methodology in ubiquitous computing environments. In: proceeding of ICCSA 2005. LNCS 3483:86–95
5. Kang BH (2007) Ubiquitous computing environment threats and defensive measures. IJMUE 2(1):47–60
6. Poslad S (2009) Ubiquitous computing: smart devices, environments and interactions. Wiley, New York, pp 3–73
7. Baldauf M, Dustdar S, Rosenberg F (2007) A survey on context-aware system. Int J Ad Hoc Ubiquit Comput 2(4):263–277
8. Android Official Blog. Google play hits 25 billion downloads. http://officialandroid.blogspot.kr/2012/09/google-play-hits-25-billion-downloads.html
9. Barkuus L, Polichar VE (2011) Empowerment through seamfulness: smart phones in everyday life. Pers Ubiquit Comput 15(6):629–639
10. Bell G, Dourish P (2007) Yesterday's tomorrows: notes on ubiquitous computing's dominant vision. Pers Ubiquit Comput 11(2):133–143
11. Campbell A, Choudhury T (2012) From smart to cognitive phones. IEEE Pervasive Comput 11(3):7–11
12. Grønli T, Chinea G, Younas M (2013) Context-aware and automatic configuration of mobile devices in cloud-enabled ubiquitous computing. Pers Ubiquit Comput
13. Ballagas R, Borchers J, Rohs M, Sheridan JG (2006) The smart phone: a ubiquitous input device. IEEE Pervasive Comput 5(1):70–77
14. Roussos G, Marsh AJ, Maglavera S (2005) Enabling pervasive computing with smart phones. IEEE Pervasive Comput 4(2):20–27
15. Orthacker C, Teufl P, Kraxberger S, Lackner G, Gissing M, Marsalek A, Leibetseder J, Prevenhueber O (2012) Android security permissions—can we trust them? In: proceeding of MOBISEC 2011. LNICST 94:40–51
16. Felt AP, Chin E, Hanna S, Song D, Wagner D (2011) Android permissions demystified. In: proceeding of CCS'11, pp 627–638
17. Felt AP, Ha E, Egelman S, Haney A, Chin E, Wagner D (2012) Android permissions: user attention, comprehension, and behavior. In: proceeding of SOUPS 2012
18. Nauman M, Khan S, Zhang X (2010) Apex: extending android permission model and enforcement with user-defined runtime constraints. In: proceeding of ASIACCS'10, pp 328–332
19. Barrera D, Kayacik H (2010) A methodology for empirical analysis of permission-based security models and its application to android. In: proceeding of CCS'10, pp 73–84
20. Zhongyang Y, Xin Z, Mao B, Xie L (2013) DroidAlarm: an all-sided static analysis tool for android privilege-escalation malware. In: proceeding of ASIACCS'13, pp 353–358
21. Bugiel S, Davi L, Dmitrienko A, Fischer T, Sadeghi A, Shastry B (2012) Towards taming privilege-escalation attacks on android. In: proceeding of NDSS 2012
22. Chin E, Felt AP, Greenwood K, Wanger D (2011) Analyzing inter-application communication in android. In: proceeding of MobiSys'11, pp 239–252
23. Wireless Geographic Logging Engine. http://wigle.net/gpsopen/gps/GPSDB/, Sep 2013
24. Gruteser M, Grunwald D (2004) A methodological assessment of location privacy risks in wireless hotspot network. In: proceeding of SPC 2003. LNCS 2802:10–24
25. Callegati F, Cerroni W, Ramilli M (2009) Man-in-the-middle attack to the HTTPS protocol. IEEE Secur Priv 7(1):78–81
26. Ariyapperuma S, Mitchell CJ (2007) Security vulnerabilities in DNS and DNSSEC. In: proceeding of ARES'07
27. Zafft A, Agu E (2012) Malicious WiFi networks: a first look. In: proceeding of SICK 2012 pp 1038–1043
28. Aime MD, Calandriello G, Lioy A, Torino PD (2012) Dependability in wireless networks: can we rely on WiFi? IEEE Secur Priv 5(1):23–29
29. Godber A, Dasgupta P (2003) Countering rogues in wireless networks. In: proceeding of ICPPW'03
30. Nikbakhsh S, Manaf ABA, Zamani M, Jangeglou M (2012) A nobel approach for rogue access point detection on the client-side. In: proceeding of WAINA'12, pp 684–687
31. Hwang H, Jung G, Sohn K, Park S (2008) A study on MITM (Man in the Middle) vulnerability in wireless network using 802.1X and EAP. In: proceeding of ICISS'08, pp 164–170
32. Georgiev M, Lyengar S, Jana S (2012) The most dangerous code in the world: validating SSL certificates in non-browser software. In: proceeding of CCS'12
33. Lee DH, Kim JG (2013) IKEv2 authentication exchange model and performance analysis in mobile IPv6 networks. Pers Ubiquit Comput

34. Wang J, Herath T, Chen R, Vishwanath A, Rao HR (2012) Phishing susceptibility: an investigation into the processing of a targeted spear phishing Email. IEEE Tran Prof Commun 55(4):345–362
35. Test application 1, https://play.google.com/store/apps/details?id=com.andromedagames.schoolrun
36. Test application 2. https://play.google.com/store/apps/details?id=air.com.cjenm.mpang.gp
37. Test application 3. https://play.google.com/store/apps/details?id=com.marvel.runjumpsmashforkakaotalk_goo
38. Test application 4. https://play.google.com/store/apps/details?id=com.pnixgames.sports
39. Test application 5. https://play.google.com/store/apps/details?id=com.cjenm.monster
40. WebView. http://developer.android.com/reference/android/webkit/WebView.html