White Rabbit in Mobile: Effect of Unsecured Clock Source in Smartphones

Shinjo Park
TU Berlin & Telekom Innovation Laboratories
pshinjo@sec.t-labs.tu-berlin.de

Altaf Shaik
TU Berlin & Telekom Innovation Laboratories
altaf329@sec.t-labs.tu-berlin.de

Ravishankar Borgaonkar
Oxford University
ravi.borgaonkar@ac.ox.ac.uk

Jean-Pierre Seifert
TU Berlin & Telekom Innovation Laboratories
jpseifert@sec.t-labs.tu-berlin.de

ABSTRACT

With its high penetration rate and relatively good clock accuracy, smartphones are replacing watches in several market segments. Modern smartphones have more than one clock source to complement each other: NITZ (Network Identity and Time Zone), NTP (Network Time Protocol), and GNSS (Global Navigation Satellite System) including GPS. NITZ information is delivered by the cellular core network, indicating the network name and clock information. NTP provides a facility to synchronize the clock with a time server. Among these clock sources, only NITZ and NTP are updated without user interaction, as location services require manual activation.

In this paper, we analyze security aspects of these clock sources and their impact on security features of modern smartphones. In particular, we investigate NITZ and NTP procedures over cellular networks (2G, 3G and 4G) and Wi-Fi communication respectively. Furthermore, we analyze several European, Asian, and American cellular networks from NITZ perspective. We identify three classes of vulnerabilities: specification issues in a cellular protocol, configurational issues in cellular network deployments, and implementation issues in different mobile OS's. We demonstrate how an attacker with low cost setup can spoof NITZ and NTP messages to cause Denial of Service attacks. Finally, we propose methods for securely synchronizing the clock on smartphones.

Keywords

baseband; NITZ; NTP; timekeeping; cellular network; clock

SPSM'16, October 24 2016, Vienna, Austria

DOI: http://dx.doi.org/10.1145/2994459.2994465

1. INTRODUCTION

The convergence towards smartphones, and the new market opened by wearables integrated several devices into them, including wrist watches. According to CNET, the popularity of smart watches negatively affected the sales of traditional wrist watches [43]. Like wrist watches, they are designed to be carried every day, providing accurate time information. While watches require manual adjustment when the battery has been changed or when there is a change regarding DST (Daylight Saving Time)[1], smartphones automatically retrieve current time information from multiple sources and propagate it to connected wearables like smart watches. Moreover, a clock mismatch or a DST adjustment is automatically corrected in the smartphone by periodic clock synchronizations. In contrast, watches require manual adjustment when the same happens.

Today's smartphones obtain clock[2] information from multiple sources and interfaces. Smartphones without Internet connectivity can use cellular network provided NITZ (Network Identity and Time Zone) information, and Internet-capable smartphones can synchronize clock information over the Internet using protocols like NTP (Network Time Protocol). Other integrated devices like GNSS (Global Navigation Satellite System, including GPS and GLONASS) are also capable of providing clock information. It is up to the smartphone manufacturers and OS (Operating System) developers' policy to utilize and assign priorities to each clock source. Besides smartphones, M2M (Machine to Machine) and IoT (Internet of Things) devices also rely on these clock sources [22].

When the "Automatic clock update" is enabled in mobile OS, it will fetch clock information from multiple sources. The way this information is fetched depends on OS and manufacturer policies, and varies among smartphones. Furthermore, it is also unclear how smartphones manage these different clock sources. NITZ is an optional message carrying network name and clock information on 2G, 3G and 4G

[1]Although a few higher-end watches are capable of automatic clock synchronization based on external signals, they are not widely used in the market.

[2]In this paper, we use the term clock referring both time and date. Time is used when usage of time is explicitly required.

cellular network. NTP utilizes the Internet connection to synchronize with a time server, and most mobile OS including Android and iOS are capable of clock synchronization via NTP. Unlike NITZ and NTP, GNSS services must be manually turned on by the user to retrieve clock information. Therefore we only discuss NITZ and NTP in this paper, as they are updated without explicitly asking the user.

In this paper, we show that it is possible to spoof clock information on modern smartphones by any third party, or an attacker with a rogue base station and Wi-Fi access point. We highlight that due to the lack of standard timekeeping methods among vendors and OS developers, unintended consequences arise on mobile OS and apps. We also discuss interesting problems with smartphones using 32-bit and 64-bit that lead to OS crashes.

To this end we make the following contributions:

- We systematically analyze security aspects provided to NITZ feature in the 3GPP (Third Generation Partnership Project) specifications of 2G, 3G, and 4G networks. The analysis indicates a lack of authentication and integrity protection in 2G networks.
- We built an experimental real network to demonstrate attacker's capabilities to spoof NITZ information and investigate their impact against both mobile and baseband OS. Furthermore, we discover NITZ configuration issues in several deployed cellular networks. We present a class of Denial of Service attacks by exploiting weaknesses in a 3GPP specification, device and cellular network configurations, and implementation issues in few baseband operating systems.
- We propose countermeasures for Denial of Service attacks and present best practices for mobile OS and app developers to maintain clock accurately.

2. RELATED WORK

We divide related work into three categories: rogue base station, NTP attack, and timekeeping on smartphones.

Rogue base station attack. A fake base station can be used to passively attack nearby mobile users. Due to specification and implementation flaws, numerous private information leakage and malicious information injection attacks were proposed. An example of an information leakage is the location leak, which Kune et al. [34] and Shaik et al. [51] discovered on 2G and 4G network respectively.

Message injection attacks using fake base stations have been proposed in several works. Mulliner et al. [42] presented SMS injection attacks, which resulted in a system crash in some cases. Weinmann [55] showed GSM control plane message injection attacks, causing various impacts. Alecu [13] proposed an SMS-based attack to exploit SIM card applications. Work of Shaik et al. [51] also contains message injection using a fake LTE base station. They mainly use messages related to location leaks. To the best of our knowledge, there are no studies evaluating risks associated with clock spoofing via a fake base station in the literature.

NTP attack. Because NTP lacks authentication and integrity protection by default, and some NTP commands could be used maliciously, both attacking NTP itself and using NTP as an attack vector are proposed. Being an UDP/IP based protocol, any generic attack like spoofing and redirecting on UDP and IP in addition to NTP-specific attacks can be performed.

There are a number of reported NTP server vulnerabilities which could be used to affect NTP availability and time accuracy, including one from Röttger [49]. Malhotra et al. [35] presented several attacks on NTP. Attacks presented in their work are targeting both the NTP server and the transmission channel to the client. Time-shifting attacks can alter any device that solely relies on NTP as a clock source. We are not focusing on the NTP attack itself. Instead, we evaluate how NTP and other clock sources are used in smartphones and how spoofing one or more of them affects the system behavior.

Klein [33] showed the impact of attacking clock sources on multiple platforms, including Android and applications on it. Czyz et al. [23] analyzed NTP amplification attacks, an attack vector utilizing NTP. CloudFlare and other online games were among the victim of this attack. Goodin [25] found this attack can cause DoS ranging up to 100Gbps. CloudFlare published details of how 400Gbps DDoS attack towards their infrastructure was possible via NTP amplification [46].

Furthermore, NTP servers could be configured to fetch clock information from either another NTP server or high-precision clocks like GPS, radio and an atomic clock. NTP servers fetching clock information directly from precise clock sources is called *Stratum 1* server, which takes a higher level in the NTP server hierarchy and is used as a clock source of lower stratum servers. Zheng et al. [57] proposed an attack on radio clock signals used by some active *Stratum 1* servers. Compared to this work, our work directly targets smartphones which receive time information from a cellular network or an NTP server.

Timekeeping on smartphones. Several crashes and bugs were discovered related to internal time keeping of mobile OS. Straley found the regression on Apple iOS devices, when the device date is set to January 1970 [53]. Kelley and Harrigan reproduced the same bug using NTP via Wi-Fi [32], using DNS spoofing on Apple's NTP server. Apple acknowledged the problem and fixed it in iOS 9.3 and 9.3.1 update [17].

3. BACKGROUND

In this section, we briefly describe different types of clock sources used by OS's in smartphones. Among those sources, we only cover NITZ and NTP in detail. Further, we discuss how each OS obtains and synchronizes the accurate time on smartphones.

3.1 Timekeeping in Smartphones

Smartphones are running two different operating systems: one on the application processor and the other on the baseband processor. Although they are interconnected via a communication stack on mobile OS (e.g. Android Radio Interface Layer), they maintain separate clocks.

The mobile OS clock could be set either by using information received from the baseband, or (Internet) data connection using NTP. This clock is used by all applications running on the OS. Due to the lack of standard methods, it is up to the OS and device manufacturers to select any of the available clock sources.

The baseband OS clock is usually set by the cellular network using NITZ information. This clock is also supplied to

mobile OS; however, it can decide whether or not to use this clock. Normally the baseband uses this clock as a timestamp for the diagnostic messages and is invisible to users.

Most OS's are keeping their clock as monotonically increasing value since a fixed epoch (e.g. 1st January 1970 on Android and iOS), and using a fixed sized variable to represent the clock value. (`time_t` on Android and iOS) Windows has different epoch and clock storage methods [37, 39].

3.2 NITZ

3GPP specifications define signaling messages for carrying NITZ information, called *MM Information* and *GMM Information* for both 2G and 3G [9] and *EMM Information* for 4G [11]. NITZ consists of time, time zone, date, and operator name. Time information in NITZ should be accurate in minute level, and the message itself has a minimum precision of one second, according to [9].

3.3 NTP

NTP [41] requires a time server to supply an accurate clock source, and communicates with the smartphone using UDP/IP. Information acquired over NTP contains date and time. Smartphones may use either public NTP pool servers [3], or OS or device vendors can operate their own NTP server.

4. SPOOFING ATTACKS VIA NITZ AND NTP

In this section, we describe the threat model and attacks against NITZ and NTP methods. Further, we conduct practical experiments to investigate configurational issues in several cellular networks. Finally, we present spoofing attacks via a rogue base station and Wi-Fi access point.

4.1 Threat Model

The attacker is capable of operating a rogue 2G, 3G or 4G base station. To achieve this, he or she has access to the radio hardware and related software to impersonate the user's serving cellular operator network. Further capabilities include the knowledge of 3GPP and NTP specifications. Additionally, attacker can also operate rogue Wi-Fi access point allowing open access to the mobile devices nearby.

4.2 Experimental Setup

Figure 1 shows our experimental setup. For NITZ, our hardware consists of a host PC and a USRP B210 device, connected via USB 3.0. The host PC runs all the software required to operate a cellular network, available from various open source projects: OpenBTS [47] for 2G, OpenBTS-UMTS [48] for 3G, and OpenLTE [56] for 4G. We modified these software stacks to operate as rogue base stations. We have selected popular smartphones from various baseband and OS manufacturers for our test purposes.

Additionally, we built a custom tool by reverse engineering a baseband to capture 2G, 3G, and 4G network messages from the test phones. In particular, we analyzed control plane messages and baseband timestamps.

For NTP, we operate a Wi-Fi access point based on OpenWRT, and a custom NTP server based on busybox-ntpd [21]. We configured access point to connect every smartphone nearby.

Figure 1: Our experimental setup, showing the USRP (left), a Wi-Fi access point (center, dotted square), and our test phones.

Ethical concerns. 2G, 3G and 4G network experiments are performed inside a faraday cage to obey to legal regulations and avoid interference with other phones. The name of the Wi-Fi network is clearly indicated as an experimental network. Further, we responsibly informed all the affected mobile OS and smartphone vendors and our reports were acknowledged by them.

4.3 Attack Background

We now discuss the security procedure defined for NITZ features in 3GPP and evaluate how cellular network operators deploy these features in practice. We also analyze the management of NITZ and NTP clock sources, and their effect on the mobile OS. We will later utilize these aspects to design our attacks.

4.3.1 Security Issues in 3GPP Specifications

Although the content of NITZ messages is similar in all the network generations, their security requirements are different and are discussed below. These security requirement means the way NITZ messages are transferred between the base station and the mobile device.

2G

The 2G specification [9, 8] defines ciphering as an optional feature, but lacks mandatory mutual authentication and integrity protection on signaling messages including NITZ. Thus, an attacker can masquerade himself as a real network operator and send spoofed NITZ information to phones.

3G

The 3G specification [6] introduced mandatory mutual authentication and integrity protection for signaling messages. Therefore, NITZ information should be processed only when a security setup is present, i.e. after authentication, integrity protection and the ciphering setup.

4G

The 4G specification [7] introduced further security enhancements, while retaining security requirement of NITZ introduced in 3G. As a result, NITZ on 4G is only processed after the authentication and establishment of integrity and ciphering, like on 3G.

Unless the attacker breaks 3G or 4G authentication or

Operator (Country)	2G/3G	4G
BASE (BE)	·	·
Mobistar (BE)	✓	✓
Proximus (BE)	✓	✓
Vodafone (DE)	·	·
E-Plus (DE)	▲	▲
Telekom (DE)	·	·
O2 (DE)	▲	·
Yoigo (ES)	✓	✓
Bouygues (FR)	✓	✓
Nova (IS)	·	·
Siminn (IS)	✓	✓
Vodafone (IS)	✓	·
NTT DoCoMo (JP)	·	▲
SK Telecom (KR)	✓	✓
KT (KR)	✓	✓
AT&T (US)	·	✓
T-Mobile (US)	✓	✓

Table 1: Operator NITZ policy. Check sign: NITZ is sent for all registration, triangle: NITZ is sent inconsistently, dot: NITZ is not sent at all.

the baseband standard is implemented incorrectly, it is impossible to spoof NITZ messages by masquerading as a real operator. We utilize implementation problems in certain basebands later.

4.3.2 *Cellular Network Operator Configuration Issues*

NITZ information is conveyed to the smartphone during several instances such as when it successfully registers to the network, moves to a different time zone and others mentioned in [10]. As sending NITZ is an optional feature, cellular operators do not send NITZ information regularly to smartphones. By using our custom tool, we analyzed real network traces during the registration process to determine which networks implement the NITZ feature.

Table 1 shows difference of NITZ configuration policies deployed by several European, Asian and US cellular operators. Operators with check sign means that the network is sending NITZ whenever the smartphone registers with it. The rows indicated by a triangle represent the network operators who do not have a consistent NITZ policy. In other words, these operators are not sending NITZ during every registration. The rows marked with a dot mean that the network operators do not send NITZ information at all.

One of the European operators stated that, since most of the smartphones today have an Internet connection, they receive their time information via NTP servers which restrains them from sending NITZ. The fact that no NITZ information is sent during registration causes clock synchronization problems on smartphones. For example, in a scenario where a user roams from T-Mobile USA (sending NITZ) to Telekom Germany (not sending NITZ) will not be able to acquire clock information automatically. Hence, sending NITZ regularly or at least during registration can avoid these problems. In the absence of NITZ, smartphones require a data connection to update the clock (via NTP). However manual updates are still possible.

Phone	NTP → NITZ	NITZ → NTP
Apple iPhone 6S	NITZ	NITZ
Google Nexus 5	NITZ	NITZ
Asus Zenfone 2E	NITZ	NITZ
HTC One E9	NITZ	NITZ
HTC One M9	NITZ	NTP
Huawei Honor 7	NITZ	NITZ
LG Vu 3	NITZ	NITZ
Samsung Galaxy Alpha	NITZ	NITZ
Samsung Galaxy S4	NITZ	NITZ
Samsung Galaxy S6	NITZ	NITZ
BlackBerry Z10	NITZ	NTP
LG Fx0	NITZ	NITZ
Microsoft Lumia 950	NITZ	NTP
Samsung Z1	NITZ	NTP

Table 2: Priorities of clock sources.

4.3.3 *NITZ vs. NTP*

Most smartphone OS's such as iOS [1], Android [28, 29], Windows Phone [36, 38][3] and Firefox OS [52] have multiple clock sources. However, there is no clear standard describing their management. Hence we performed an experiment using our setup in Section 4.2 to reveal the myths about the management and priorities of NITZ and NTP in the test smartphones.

We set up our test smartphones to retrieve clock information from the base station, followed by the Wi-Fi access point, and vice versa. We configured the base station and NTP server to send different clock values to check which clock is preferred. Our results are summarized in Table 2.

Smartphones like the Nexus 5 were reluctant to update the clock received over NTP when the clock via NITZ is already present and was received within the last 24 hours. In other words, an NTP request is only triggered once in 24 hours. This behavior is defined in the NTP handling code of the Android OS [29].

In contrast, certain smartphones are treating NITZ and NTP with equivalent priority, setting the system clock with the latest received information. For example, the HTC One M9 updates the clock via NTP in spite of having recent NITZ information. One of the reasons could be the customized HTC Sense[4] environment on Android OS. Although the Lumia 950 also falls in this category, we found that sometimes it resets to the accurate clock even when both NITZ and NTP are inaccurate. We assume that this is caused by another clock source which we do not discuss in this paper.

Apple devices running iOS version 9.3.2 behaved differently from most of smartphones. While older version of iOS preferred NITZ than NTP, newer iOS version changed timekeeping method. It neither accepts new NITZ information, nor issues an NTP request for certain amount of time after the clock is set during initial setup or periodic time update.

To summarize, NITZ is given higher priority in most cases since it is received in a secure authenticated channel and can be trusted more than NTP.

[3]Prior to Windows Phone 8.1 Update 1 it lacked NTP
[4]HTC customized Android system

OS (Architecture)	time_t Size	End Year
iOS (32-bit) [15]	32-bit	2038
iOS (64-bit)	64-bit	Never
Android (32-bit)[5]	32-bit	2038
Android (64-bit)	64-bit	Never
BlackBerry 10 [18]	Unsigned 32-bit	2106
Windows [39]	Custom	30827

Table 3: Timekeeping method of mobile OS.

Mobile OS	Default NTP Server
Android	[0-3].android.pool.ntp.org
BlackBerry 10	time.blackberry.com
Firefox OS	[0-3].pool.ntp.org
iOS	time-ios.apple.com
Sailfish OS	[0-3].sailfishos.pool.ntp.org
Tizen	pool.ntp.org
Windows	time.windows.com

Table 4: Default NTP address for each mobile OS's.

4.3.4 *Mobile OS issues*

Mobile OS's are using different ways to represent internal clock information as stated in Table 3. For example, 32-bit iOS and Android devices are only able to represent dates up to January 2038 ($2^{31} - 1$ seconds after UNIX epoch). Any future date causes a clock overflow, and will roll back to the past. Unintended consequences could happen.

Based on our experiments, sending a clock value (some seconds before 2038-01-19 03:14:07 UTC) via NITZ causes the mobile OS to crash on all 32-bit Android smartphones. Using NTP to set that clock is not always possible because NTP up to version 3 [40] can represent up to year 2036, and most smartphones utilize NTP version 3. Additionally, 32-bit iPhones like iPhone 5 also suffer from this problem [4]. Other 32-bit mobile OS like Firefox OS only rolled back its internal clock without crashing. However, the clock on 64-bit smartphones virtually never expires, as $2^{63} - 1$ seconds are far beyond the age of the universe.

Baseband OS keeps its time separate from the mobile OS. By using our custom tool we analyzed the baseband timestamps before and after sending a NITZ message. The Qualcomm basebands has a wider date range than year 2038, but it is not passed to Android due to limitations in the radio communication stack. Other basebands like Samsung or Intel, do not include clock information in debugging messages.

4.4 Spoofing Attacks

In this section, we exploit 3GPP specifications and cellular operator configuration issues to demonstrate clock spoofing attacks in two ways: via a rogue base station and via a Wi-Fi access point.

4.4.1 *Via Rogue Base Station*

We initially consider that a user's smartphone is attached to a legitimate base station. Then attacker forces the phone to attach to the rogue one (2G/3G/4G), by impersonating the user's service provider. Upon detecting a new base station the smartphone initiates a connection to it. As a response the attacker sends spoofed clock information via a *MM/GMM/EMM Information* message.

On 2G, all our test smartphones accepted spoofed NITZ information and changed their baseband clock. For 3G and 4G, certain smartphones accepted NITZ and changed the baseband clock in the absence of authentication, thereby violating 3GPP specifications. The received clock information is forwarded to the mobile OS and used by apps.

Unlike other smartphones, Samsung smartphones (Galaxy Alpha, Galaxy S4, S6, Z1) are changing the time zone of it upon receiving broadcast information such as MCC (Mobile Country Code) and MNC (Mobile Network Code) from a base station before connecting to it. For example, based on the MCC and MNC of Vietnam, the time zone is updated to Indochina Time (UTC+07:00). This behavior is unreliable due to the fact that broadcast messages are accepted by the smartphone without having any security setup.

As shown in Section 4.3.3, after detaching from the rogue base station the smartphone may or may not receive accurate clock information via NITZ depending on the operator policy. Also, an NTP update is unlikely due to the preference of NITZ over NTP. As a result, we discover that to get accurate clock some smartphones required a reboot, in some cases even by toggling the automatic clock update feature.

4.4.2 *Via Wi-Fi Access Point*

Like a rogue base station, attacker can operate a rogue Wi-Fi hotspot with the same name (SSID; Service Set ID) as a legitimate one (e.g. public Wi-Fi). Upon discovery, the smartphone connects to the rogue one and all Internet traffic will be routed via the attacker.

We configured our Wi-Fi access point to redirect all packets towards UDP port 123 (default NTP port) to our rogue NTP server. Upon receiving NTP requests from the smartphone, the NTP server replied with spoofed clock information. Because of the wide varieties of NTP server addresses among mobile OS's as shown in Table 4, we use a port redirection instead of DNS query spoofing for simplicity and scalability of the attack. Since NTP does not mandate any authentication, all our tested smartphones accepted spoofed clock information.

Android has an NTP polling interval [29, 27] of one day by default. Upon reception of the first NTP response packet, Android will not issue further NTP requests until the NTP polling interval had been rolled back. This is also true when the user changes to the legitimate Wi-Fi network, making recovery via NTP impossible at least for one day.

5. IMPACTS ON MOBILE OS AND APPS

In this section, we discuss how clock spoofing attacks affect the operation of baseband and mobile OS and its apps. We also examine the persistence of attacks and possible recovery methods.

5.1 Baseband Operations

We analyzed control plane messages in all 2G, 3G, 4G networks [9, 11] and found that only a small number of messages contain current baseband or network clock information. As a result, most cellular operations are not affected by the attack. Other usage of the clock information on the baseband is logging its diagnostic messages.

Signaling messages for telephone calls, both circuit switch

[5]including other Linux-based mobile OS, e.g. Firefox OS, Sailfish OS, Tizen

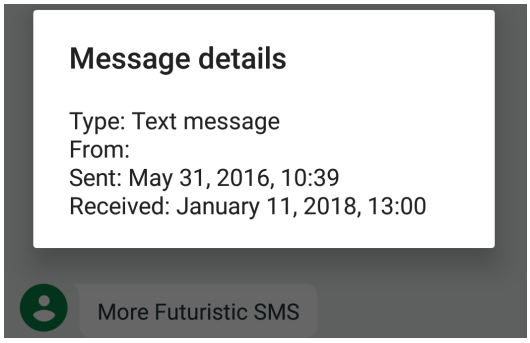

Figure 2: Message details on Nexus 5, showing both network and device timestamp.

based and VoLTE [31], do not contain clock information. As a result, incoming and outgoing calls are logged on the phone with the mobile OS clock and operators maintain their own clock for call accounting.

Incoming SMS is one of the signaling messages containing network clock information when the message has been sent [12]. It is up to the device developers to use the timestamp present in the incoming SMS or not. Figure 2 shows an example of clock difference: network clock information is included in the "Sent" field, the device counterpart is in the "Received" field.

5.2 Mobile OS Operations

Clock spoofing can affect various components, ranging from low level components like kernel and system programs to higher level components like apps.

5.2.1 *System Components*

Any system component or app relying on clock information will be vulnerable to spoofing attacks. Specifically, when the attacker sends a date around year 2038 a memory overflow occurs causing a complete system crash. Further, networking protocols such as TLS/SSL rely on accurate clock information [35].

We concentrate on TLS/SSL as it forms a basis of higher level protocols like HTTPS, and is widely used in mobile environments. Once a secure connection is established using TLS, the client performs a server certificate validity period check. If the current clock is outside of the validity period, the certificate is considered to be invalid. Depending on the type of app, the secure connection gets terminated or established nonetheless. We found that Android [30] and iOS [16] enforce strict validity checks on TLS connections, causing a Denial of Service on the smartphone when the clock is incorrect.

5.2.2 *Apps*

We studied the behavior of apps after the clock spoofing on Android and iOS. Apps not utilizing clock information or apps utilizing clock information locally are excluded from our analysis, because the former is not affected at all and tha later has no means of verifying accuracy of the clock. However, apps utilizing clock information and require a data connection are affected by the attack. We tested some of the widely used apps and noticed three different behaviors: explicitly noticing clock inaccuracy, showing a generic error

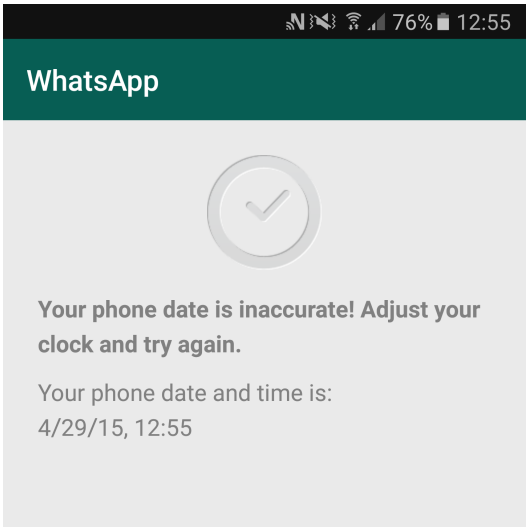

Figure 3: Best practice: screenshot of WhatsApp indicating system clock spoofing.

message, and operate regardless of the clock spoofing.

Web browser, social networking, mobile messengers mainly rely on TLS-based encryption. Specifically, WhatsApp (see Figure 3) and Google Chrome explicitly indicated the system clock mismatch when clock spoofing was detected. In this case, WhatsApp failed to operate normally until the clock was reset to the accurate value.

Further, certain apps like OpenVPN did not distinguish between clock spoofing and generic network errors. Figure 4 shows how OpenVPN presents an error when the current system clock is outside of the certificate validity period. App stores like the Google Play Store presented an option to retry the connection, but this fails to work until the clock is reset.

Finally, certain apps operated regardless of clock spoofing and showed no indication to the user. We observed this behavior on apps handling mostly public information like public transportation timetable, etc. Samsung Knox [50] also falls into this category, as it allows unlocking and locking the secure container regardless of clock spoofing.

5.3 Persistence and Recovery

We observed that majority of our test smartphones are affected by the attack, and failed to receive accurate clock information even after the attacker shuts down the rogue base station or Wi-Fi access point. During this state, mobile apps failed to operate normally, causing a persistent Denial of Service on the smartphone.

In order to recover, the user needs to reboot the smartphone. In some cases it is enough to toggle the flight mode. When the user is still in range of the rogue base station or Wi-Fi access point, rebooting will likely pick up false clock information again and cause apps to fail. Similarly, when traveling accross time zone, rebooting the smartphone is usually suggested when current clock information is incorrect [14].

Rebooting the smartphone would also be required after a system crash caused by the year 2038 problem. Depending on the smartphone, the clock resets back to the UNIX time epoch (year 1970), or before the epoch (year 1901) until receiving accurate clock information.

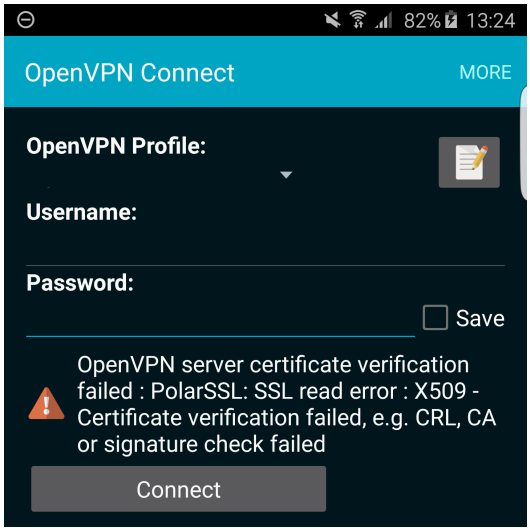

Figure 4: Common practice: screenshot of OpenVPN indicating network failure.

6. DISCUSSION

We now analyze the security implications that arise due to inadequate timekeeping methods adopted by smartphone and baseband manufacturers, and mobile OS developers. Further, we propose effective mechanisms to handle clock information inside each component of smartphones.

Baseband manufacturers. As the mobile OS prefers to receive clock information over NITZ assuming that it is acquired over a secure channel, baseband manufacturers should abide by the security measures already implemented in respective specifications for NITZ.

Moreover, as operating a rogue base station now became easier [20], baseband manufacturers are required to apply defensive programming regarding the data received from the network provider. An example of defensive programming would be to verify the received NITZ values. The 3GPP specification does not clearly define error handling procedures when NITZ information contains an invalid clock value [9, 12]. As a result, when the base station sends month 13, it could be interpreted as next year January or could be ignored. Likewise, hour 25 could be interpreted as 01:00 next day. All these behaviors are accepted. Some baseband accepted out-of-range values and interpreted them as a time in the future, while some ignored those values. As the clock information first arrives on the baseband processor, a sanity check will release the burden of verifying the data inside the mobile OS.

Mobile OS developers. OS need to provide consistent and predictable policies on clock updates. As clock sources have different availability and security, more available and secure clock sources should always be preferred. Also, mobile OS can provide interfaces to trigger manual updates of the clock, and a minimal notification of the current clock source when the clock source had been changed. For example, BlackBerry has an option to ask the user about time zone updates when a new time zone is detected [19]. This can save the user from rebooting the phone when traveling across time zones, and gives user the chance to use a manual clock when some of the clock sources are not working.

Recent Apple iOS 9.3 series introduced some of the sanitization mentioned in Section 4.3.3. Contrary to iOS, our test Android smartphones changed its clock when there were a large clock drift. Although Android has some NITZ sanitization procedures [28] but it only checks system uptime upon receiving NITZ, not the content of NITZ information. As a result, mobile OS clock changed in cases when we sent a correct time value first (e.g. 1st July midnight), then a time value of some days in the future or past (e.g. 10th July or 20th June), and then again recovered the original time (e.g. 1st July 00:10).

Although iPhone 5s opened the era of 64-bit smartphones in July 2013, a large number of 32-bit smartphones and cellular capable M2M and IoT devices will remain in operation for an extended amount of time. Therefore, OS developers should be aware of the year 2038 problem and need to provide fallback mechanisms without causing the OS to crash. Although the problem itself was first reported on Android 5 years ago [2], the bug was virtually abandoned and remained hidden for a long time.

Additionally, OS developers and/or device manufacturers can implement secure NTP to protect users from NTP spoofing attacks. NTP version 4 introduced cryptographic authentication features, and there are studies proposing secure NTP [24]. Authenticated NTP is already in operation by NIST [44] and the US Naval Observatory [45]. Some device manufacturers like Apple and BlackBerry are operating NTP servers themselves, and actively utilizing secure containers in their respective mobile OS. They can provide NTP with authentication as a premium service.

App developers. Among our set of test apps, WhatsApp and Google Chrome showed the best way to handle the clock spoofing. Other apps using TLS did not distinguish correctly between errors caused by clock spoofing from generic network errors. Unlike other network errors, errors caused by clock spoofing are easier to detect and recover from by the user. We therefore suggest app developers to distinguish clock spoofing errors from other errors, unless the app is not utilizing clock information.

Also, if clock information is extensively used inside an app, operating a separate time server on the app developer's side can help making accurate timekeeping less of an issue inside the application. Some time-critical apps like banking apps are already following this.

App developers can differentiate certificate validity period, compromising between security and maintenance cost. Google set their certificate validity to 3 months, while Facebook and Twitter used 2 years. As a result, Google apps are stricter on clock spoofing attacks as compared to Facebook and Twitter.

7. CONCLUDING REMARKS AND FUTURE WORK

Smartphones are utilizing multiple clock sources via cellular network and Internet to ensure accurate clock information. We have shown that the vulnerabilities we discovered in cellular network (2G, 3G and 4G), NTP standards, and implementations allowed clock spoofing attack towards smartphone users. Our attacks were performed using open source cellular network and NTP server software, with readily available hardware and several smartphones with 6 different mobile OS's. We demonstrated Denial of Service attack

leading to the malfunction of apps and complete mobile OS crash. Baseband implementations, 3GPP standard issues, and lack of consistent timekeeping methods in smartphones are the root causes of these attacks. We proposed best practices to manage multiple clock sources in smartphones and provide accurate clock to the user. We followed standard responsible disclosure methods of all affected manufacturers, and we also notified the mobile app developers and OS developers. Google addressed the problem mentioned in this paper in Android Security Bulletin, August 2016 [26, 5].

Our future work will cover the other clock sources we excluded or did not analyzed in this paper, and the other components of mobile OS and apps. Although we explicitly excluded GPS in this paper, GPS spoofing including clock information is already proposed [54]. While previous work [35] mentioned that multiple protocols are affected by clock spoofing attack, we only covered TLS/SSL in this paper. We will also cover other network protocols and OS components that is largely affected by clock spoofing attack.

We belive our attacks are not only true for smartphones but also for IoT systems that derive from a mobile OS (for example, connected cars). There is more protection and reliability needed for maintaining an accurate time information for these systems. Our research results exhibit that clock spoofing attacks needs more attention in the age of digitally connected world.

Acknowledgements

This research was partly performed within the 5G-ENSURE project (www.5GEnsure.eu) the EU Framework Programme for Research and Innovation Horizon 2020 under grant agreement no. 671562 and received funding from the Software Campus project from DLR (Deutsches Zentrum für Luft- und Raumfahrt) project no. 01IS12056. We would like to thank Kibum Choi, Byeongdo Hong, Dongkwan Kim, Hongil Kim and Youngseok Park from KAIST on collection of data, and the anonymous reviewers for their thoughtful feedback on previous versions of this paper.

8. REFERENCES

[1] Carrier.plist - The iPhone Wiki. https://www.theiphonewiki.com/wiki/Carrier.plist.

[2] Issue 16899: Year 2038 problem - Android Open Source Project Issue Tracker. https://code.google.com/p/android/issues/detail?id=16899.

[3] pool.ntp.org project : the internet cluster of ntp servers. http://www.pool.ntp.org/en/.

[4] The iPhone Apocalypse: January 19, 2038 - MacRumors Forums. http://forums.macrumors.com/threads/the-iphone-apocalypse-january-19-2038.1943912/.

[5] CVE-2016-3831. Available from MITRE, CVE-ID CVE-2016-3831., 2016.

[6] 3GPP. 3G security; Security architecture. TS 33.102, 3rd Generation Partnership Project (3GPP).

[7] 3GPP. 3GPP System Architecture Evolution (SAE); Security architecture. TS 33.401, 3rd Generation Partnership Project (3GPP).

[8] 3GPP. Digital cellular telecommunications system (Phase 2+); Security aspects. TS 42.009, 3rd Generation Partnership Project (3GPP).

[9] 3GPP. Mobile radio interface Layer 3 specification; Core network protocols; Stage 3. TS 24.008, 3rd Generation Partnership Project (3GPP).

[10] 3GPP. Network Identity and TimeZone (NITZ); Service description; Stage 1. TS 22.042, 3rd Generation Partnership Project (3GPP).

[11] 3GPP. Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS); Stage 3. TS 24.301, 3rd Generation Partnership Project (3GPP).

[12] 3GPP. Technical realization of the Short Message Service (SMS). TS 23.040, 3rd Generation Partnership Project (3GPP).

[13] B. Alecu. SMS Fuzzing - SIM Toolkit Attack. *DEF CON 21*, 2013.

[14] AndroidCentral.com. Automatic time zone and date/clock are wrong. http://forums.androidcentral.com/htc-one-m7/291916-automatic-time-zone-date-clock-wrong.html.

[15] Apple Inc. Major 64-Bit Changes - iOS Developer Library. https://developer.apple.com/library/ios/documentation/General/Conceptual/CocoaTouch64BitGuide/Major64-BitChanges/Major64-BitChanges.html.

[16] Apple Inc. Using Network Securely - iOS Developer Library. https://developer.apple.com/library/ios/documentation/NetworkingInternetWeb/Conceptual/NetworkingOverview/SecureNetworking/SecureNetworking.html.

[17] Apple Support. If you changed the date to May 1970 or earlier and can't restart your iPhone, iPad, or iPod touch. https://support.apple.com/en-us/HT205248.

[18] BlackBerry Limited. Clock and timer services - Native SDK for BlackBerry 10. https://developer.blackberry.com/native/documentation/dev/rtos/arch/kernel_clockandtimer.html.

[19] BlackBerry Limited. When traveling between time zones the BlackBerry smartphone does not automatically update to local time. http://support.blackberry.com/kb/articleDetail?ArticleNumber=000010323.

[20] R. Borgaonkar, K. Redon, and J.-P. Seifert. Security analysis of a femtocell device. In *Proceedings of the 4th International Conference on Security of Information and Networks*, SIN '11, pages 95–102, New York, NY, USA, 2011. ACM.

[21] Bruce Perens et al. BusyBox. https://busybox.net/about.html.

[22] Cinterion Wireless Modules. BGS2-E AT Command Specification.

[23] J. Czyz, M. Kallitsis, M. Gharaibeh, C. Papadopoulos, M. Bailey, and M. Karir. Taming the 800 Pound Gorilla: The Rise and Decline of NTP DDoS Attacks. In *Proceedings of the 2014 Conference on Internet Measurement Conference*, IMC '14. ACM, 2014.

[24] B. Dowling, D. Stebila, and G. Zaverucha. Authenticated network time synchronization. *IACR Cryptology ePrint Archive*, 2015:171, 2015.

[25] D. Goodin. New DoS attacks taking down game sites deliver crippling 100Gbps floods. http://arstechnica.com/security/2014/01/new-dos-attacks-taking-down-game-sites-deliver-crippling-100-

gbps-floods/.

[26] Google Inc. Android Security Bulletin–August 2016. https://source.android.com/security/bulletin/2016-08-01.html.

[27] Google Inc. config.xml. https://android.googlesource.com/platform/frameworks/base/+/master/core/res/res/values/config.xml.

[28] Google Inc. GsmServiceStateTracker.java. https://android.googlesource.com/platform/frameworks/opt/telephony/+/master/src/java/com/android/internal/telephony/gsm/GsmServiceStateTracker.java.

[29] Google Inc. NetworkTimeUpdateService.java. https://android.googlesource.com/platform/frameworks/base.git/+/master/services/core/java/com/android/server/NetworkTimeUpdateService.java.

[30] Google Inc. Security with HTTPS and SSL - Android Developers. http://developer.android.com/training/articles/security-ssl.html.

[31] GSMA. Official Document IR.92 - IMS Profile for Voice and SMS.

[32] P. Kelley and M. Harrigan. iOS 1970 Vulnerability Leads to Remote Bricking of Phones Over The Air. https://www.youtube.com/watch?v=zivWTwOjEME.

[33] J. Klein. Becoming a Time Lord: Implications of Attacking Time Sources. *Shmoocon FireTalks 2013*, 2013.

[34] D. F. Kune, J. Kölndorfer, N. Hopper, and Y. Kim. Location leaks over the GSM air interface. *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*, 2012.

[35] A. Malhotra, I. E. Cohen, E. Brakke, and S. Goldberg. Attacking the Network Time Protocol. In *23rd Annual Network and Distributed System Security Symposium, NDSS 2016*, 2016.

[36] Microsoft. EnableAutomaticTime - Windows 10 hardware dev. https://msdn.microsoft.com/en-us/library/windows/hardware/mt502640(v=vs.85).aspx.

[37] Microsoft. FILETIME structure - Windows Dev Center. https://msdn.microsoft.com/en-us/library/windows/desktop/ms724284%28v=vs.85%29.aspx.

[38] Microsoft. NTPEnabled - Windows 10 hardware dev. https://msdn.microsoft.com/en-us/library/windows/hardware/mt157027(v=vs.85).aspx.

[39] Microsoft. SYSTEMTIME structure - Windows Dev Center. https://msdn.microsoft.com/en-us/library/windows/desktop/ms724950%28v=vs.85%29.aspx.

[40] D. Mills. Network Time Protocol (Version 3) Specification, Implementation and Analysis. RFC 1305 (Draft Standard), Mar. 1992. Obsoleted by RFC 5905.

[41] D. Mills, J. Martin, J. Burbank, and W. Kasch. Network Time Protocol Version 4: Protocol and Algorithms Specification. RFC 5905 (Proposed Standard), June 2010.

[42] C. Mulliner, N. Golde, and J.-P. Seifert. SMS of Death: from analyzing to attacking mobile phones on a large scale. *USENIX Security*, 2011.

[43] S. Musil. Smartwatches now more popular than Swiss watches, thanks to Apple. *CNET*, 2016.

[44] National Institute of Standards and Technology. The NIST Authenticated NTP Service. http://www.nist.gov/pml/div688/grp40/auth-ntp.cfm.

[45] Naval Meteorology and Oceanography Command. Authenticated NTP - DoD Customers. http://www.usno.navy.mil/USNO/time/ntp/dod-customers.

[46] M. Prince. Technical Details Behind a 400Gbps NTP Amplification DDoS Attack. https://blog.cloudflare.com/technical-details-behind-a-400gbps-ntp-amplification-ddos-attack/.

[47] Range Networks. OpenBTS. http://openbts.org/.

[48] Range Networks. OpenBTS-UMTS. http://openbts.org/w/index.php?title=OpenBTS-UMTS.

[49] S. Röttger. Finding and exploiting ntpd vulnerabilities. http://googleprojectzero.blogspot.de/2015/01/finding-and-exploiting-ntpd.html.

[50] Samsung. Mobile Enterprise Security – Samsung Knox. https://www.samsungknox.com/en.

[51] A. Shaik, R. Borgaonkar, N. Asokan, V. Niemi, and J.-P. Seifert. Practical attacks against privacy and availability in 4G/LTE mobile communication systems. In *23rd Annual Network and Distributed System Security Symposium, NDSS San Diego, California, USA, February 21-24, 2016*, 2016.

[52] Shao Hang Kao. Bug 874771 - Implement SNTP support (Gecko end). https://hg.mozilla.org/mozilla-central/rev/fb5ba2a8e039.

[53] Z. Straley. Don't set your iPhone's date to January 1, 1970! The fastest trick to BRICK an iPhone! https://www.youtube.com/watch?v=fY-ahR1R6IE.

[54] N. O. Tippenhauer, C. Pöpper, K. B. Rasmussen, and S. Capkun. On the Requirements for Successful GPS Spoofing Attacks. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS, 2011.

[55] R.-P. Weinmann. Baseband Attacks: Remote Exploitation of Memory Corruptions in Cellular Protocol Stacks. *USENIX Workshop on Offensive Technologies*, 2012.

[56] B. Wojtowicz. OpenLTE. https://sourceforge.net/projects/openlte/.

[57] Y. Zheng and H. Shan. Time is On My Side: Forging a Wireless Time Signal to Attack NTP Servers. *HITB 2016 Amsterdam*, 2016.