# Time and Position Spoofing with Open Source Projects

Kang Wang
Mobile Security of Alibaba Group
wangkang.wk@alibaba-inc.com

Shuhua Chen
Mobile Security of Alibaba Group
xingzhong@alibaba-inc.com

Aimin Pan
Mobile Security of Alibaba Group
aimin.pan@alibaba-inc.com

***Abstract*—Time and position data of mobile devices are trusted without checking by most vendors and developers. We discover a method of GPS spoofing with low-cost SDR devices. The method can be used to alter the location status as well as the time of affected devices, which poses a security threat to location-based services. We also examine other positioning methods used by smart devices (e.g. WiFi) and how to spoof them. Advices on preventing such spoofing are given.**

## I. Introduction

GPS is widely used for positioning and time syncronization for mobile devices. However, since time and position data of mobile devices are trusted and seldom verified by most vendors and developers, it provides a huge attack surface for potential attackers.

In this paper, we demonstrate that time and position data of mobile devices can be easily cheated using open source tools, and neither physical touch with mobile devices nor jailbreak/-root process is necessary. It is able to interfere all the position and time of cellphones in the surrounding area. Several years ago, it is still very expensive for personal potential attackers to obtain SDR devices. But with SDR platforms becoming much cheaper, hardware cost of this method is only about $300 today and could be even cheaper later.

First, a method of GPS signal spoofing with SDR platform is introduced and demonstrated. A software defined radio platform is required to complete this attack. Faked GPS baseband signal samples are generated using an open source project, and replayed using SDR platform such as HackRF, BladeRF or USRP. The time of affected devices is altered as well.

Second, a method of WiFi based positioning spoofing is introduced and demonstrated. It only requires a Linux laptop with a wireless adapter. A demonstration of cheating Baidu Map app with faked WiFi SSIDs and BSSIDs is shown. Tools and scripts to complete this attack are given is this section as well.

In our opinion, it is urgent to inform vendors and developers of mobile devices that GPS and WiFi positioning data are not reliable, and should be verified carefully. In the end, some advices on preventing such spoofing are provided.

The rest of this paper is organized as follows. Section II first gives an overview on GPS system including positioning principle, GPS signal frames and GPS broadcast ephemeris, then introduces the experiment platform and software, and finally shows some examples of GPS spoofing results. Section III introduces WiFi based location spoofing principle, then gives the detailed method of this spoofing, and finally shows a few examples. Section IV gives some advices on how to prevent such spoofing.

## II. GPS Spoofing

### A. GPS Overview

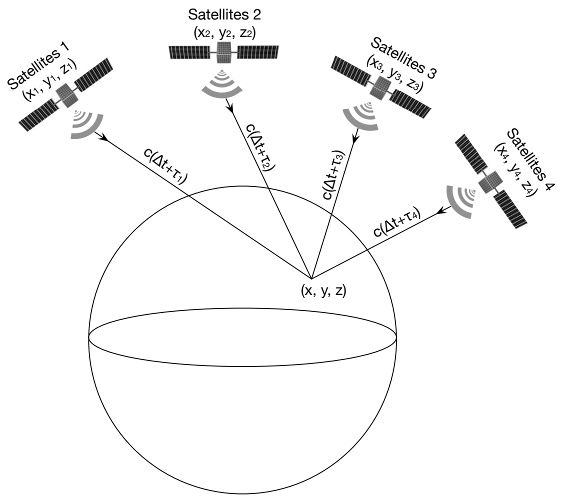


Fig. 1. GPS Positioning Principle

*1) GPS Positioning Principle:* First, let us clarify the problem. What we want to know is our position coordinate $(x, y, z)$. If there is a position $A$ (in fact, it's a satellite) which has a known coordinate $(x_1, y_1, z_1)$, we can simply broadcast a signal, which can be either light or sound or electromagnetic wave, from position $A$. Then we try to measure the duration $\tau_1$ between the signal is sent and arrived. In GPS system, we choose electromagnetic wave as the signal, so we know the velocity of electromagnetic wave, $c$. And we can give this equation as below:

$$\sqrt{(x-x_1)^2+(y-y_1)^2+(z-z_1)^2} = c\tau_1$$

This equation cannot be solved since it has three unknowns. So we can simply add another two positions(satellites) whose coordinates are already known too. Let's mark them as

$(x_2, y_2, z_2)$ and $(x_3, y_3, z_3)$. And we also measure the durations $\tau_2$ and $\tau_3$. So we can get an equation set as below:

$$\sqrt{(x-x_1)^2+(y-y_1)^2+(z-z_1)^2} = c\tau_1$$
$$\sqrt{(x-x_2)^2+(y-y_2)^2+(z-z_2)^2} = c\tau_2$$
$$\sqrt{(x-x_3)^2+(y-y_3)^2+(z-z_3)^2} = c\tau_3$$

Now, our position coordinate $(x, y, z)$ can be solved out of the equations.

But it is not enough in engineering practice. For the measurement of duration between electromagnetic wave is sent and arrived, $\tau_1$, a timestamp $t_1$ is needed to write into this electromagnetic wave signal just the same time as it is sent from the satellite, and the time reference of which is from the clock carried in the satellite. When the signal arrives at our position, we extract the timestamp $t_1$ from the signal, then calculate the time offset between $t_1$ and localtime $t_2$ to get the duration $\tau_1$. But local clock and satellite clock are not synchronized, there is a time deviation $\Delta t_1$ between them. This deviation brought by clock desynchronization should be taken into consideration. The revised equations are as below:

$$\sqrt{(x-x_i)^2+(y-y_i)^2+(z-z_i)^2} = c(\Delta t_i+\tau_i), i \in \{1,2,3\}$$

This equation set is unsolvable again, since there are another three unknowns, $\Delta t_1$, $\Delta t_2$, $\Delta t_3$.

The clocks carried by each GPS satellites are high precision atomic clocks, which are in strict synchronization state. So we have the following equation:

$$\Delta t_1 = \Delta t_2 = \Delta t_3 = \Delta t$$

Then the equation set becomes:

$$\sqrt{(x-x_i)^2+(y-y_i)^2+(z-z_i)^2} = c(\Delta t+\tau_i), i \in \{1,2,3\}$$

Still, we cannot solve this equation set for now. So we add the 4th satellite. The equation set becomes:

$$\sqrt{(x-x_1)^2+(y-y_1)^2+(z-z_1)^2} = c(\Delta t+\tau_1)$$
$$\sqrt{(x-x_2)^2+(y-y_2)^2+(z-z_2)^2} = c(\Delta t+\tau_2)$$
$$\sqrt{(x-x_3)^2+(y-y_3)^2+(z-z_3)^2} = c(\Delta t+\tau_3)$$
$$\sqrt{(x-x_4)^2+(y-y_4)^2+(z-z_4)^2} = c(\Delta t+\tau_4)$$

Where

- $(x, y, z)$ is the coordinate of receiver antenna.
- $(x_i, y_i, z_i)$ is the coordinate of the $i$th GPS satellite.
- $c$ is the speed of light.
- $c(\Delta t + \tau_i)$ is the distance from the receiver antenna to the satellite antenna including receiver and satellite clock offsets (and other biases, such as atmospheric delays), a.k.a the pseudo-range (PR).
- $\tau_i$ is the signal travel duration

Now that our position $(x, y, z)$ can be solved out of this equation set. That is why at least 4 satellites are needed to complete the GPS positioning. Besides, we can also calculate our local clock offset against the atomic clock in GPS satellites, and this procedure is known as GPS time synchronization.

TABLE I
GPS L1 SIGNAL

| Parameter | Value |
|---|---|
| Code | C/A Code |
| Modulation | BPSK |
| Frequency | 1575.42MHz |
| Code Rate | 1.023 MHz |

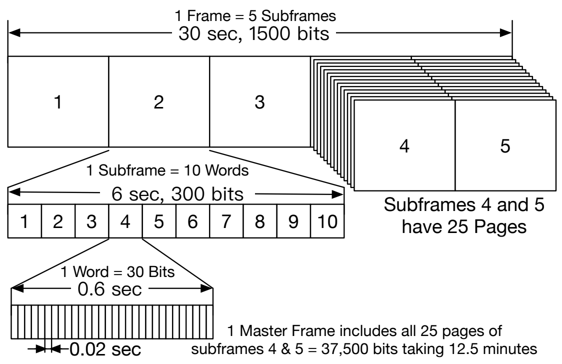


Fig. 2. GPS Signal Frame

*2) GPS Frames:* GPS signal frame structure is shown in Fig. 2. The bitrate of GPS signal is 50 bps. GPS satellites broadcast GPS signals in different frequency band and in different modulation. L1 signal is the most common signal in civil usage.

The strength of GPS signal received is very weak, at about -130 dBm, and most GPS receivers wouldn't work indoor. It makes GPS signal interference or spoofing quite easy, since attackers don't need to generate a strong signal to cover the real GPS signal.

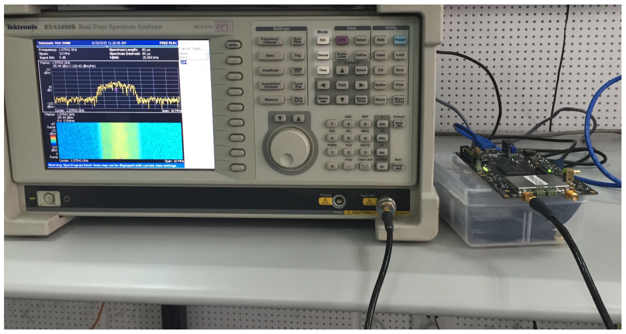


Fig. 3. Generated GPS Signal Spectrum

*3) BRDC Data:* BRDC (Broadcast Ephemeris Data) files contain the unique GPS satellite ephemeris messages for each day. Ephemeris data provides the exact location data $(x_i(t), y_i(t), z_i(t))$ of each satellites, so that receivers can get prior information in order to calculate position.

You can download BRDC archives in form of RINEX (Receiver Independent Exchange Format) from ftp://cddis.gsfc.nasa.gov/gnss/data/daily/.

The archives are named in the following scheme by rules in Table II.

```
YYYY/DDD/YYn/brdcDDD0.YYn.Z
```

TABLE II
BRDC FILENAME RULES

| Code | Meaning |
|---|---|
| YYYY | 4-digit year |
| YY | 2-digit year |
| DDD | 3-digit day of year |
| .Z | compressed Unix file |

For example, ‘brdc3540.14n’ means GPS satellite ephemeris messages of December, 20th, 2014.

## B. GPS Position Spoofing Using Opensource Code and SDR

*1) GPS-SDR-SIM Project:* GPS-SDR-SIM project is an opensource GPS baseband signal generator released under MIT license. The principal author of this project is Takuji Ebinuma. We have contributed code of static location support and BladeRF script support to this project.

It takes a RINEX formatted GPS ephemeris archive and location as input, and generates GPS baseband signal for SDR platform to playback. The default maximum duration of samples generated is three minutes.

```
$ git clone git@github.com:osqzss/gps-sdr-sim.git
$ cd gps-sdr-sim
$ gcc gpssim.c -lm -fopenmp -o gps-sdr-sim
```

Listing 1. Fetch GPS-SDR-SIM Code and Compile

```
$ ./gps-sdr-sim -h
Usage: gps-sdr-sim [options]
Options:
  -e <gps_nav>     RINEX navigation file for GPS
    ephemerides (required)
  -u <user_motion> User motion file (dynamic mode)
  -g <nmea_gga>    NMEA GGA stream (dynamic mode)
  -l <location>    Lat,Lon,Hgt (static mode) e.g.
    30.286502,120.032669,100
  -o <output>      I/Q sampling data file (default:
    gpssim.bin)
  -s <frequency>   Sampling frequency [Hz] (default:
     2600000)
  -b <iq_bits>     I/Q data format [8/16] (default:
    8)
```

Listing 2. Usage of GPS-SDR-SIM

*2) SDR Platforms:* To transmit signals into real air, we need Software Defined Radio platform such as HackRF, BladeRF, USRP.

HackRF is an open source SDR platform using USB 2.0 interface, operating on frequency band from 10MHz to 6GHz, supporting 20Msps sample rate maximum, but only half duplex. It provides every design details from host driver to ARM firmware and even hardware scheme, PCB layout and BOM. BladeRF uses USB 3.0 interface, operating on frequency band from 300MHz to 3.8GHz, supporting independent RX/TX 12-bit 40MSPS quadrature sampling and full duplex.

BladeRF seems to have a better radio frequency performance than HackRF, since it uses an integral radio chip to support all working RF frequency range from 300MHz to 3.8GHz. HackRF uses some RF switches and RF converters on PCB board to expand a narrow working frequency between 2.3GHz and 2.7GHz to 10MHz and 6GHz, which brings quite a lot of RF power loss. But according to our experiment, both GPS signal transmitted by HackRF and BladeRF can be decoded by regular GPS receivers.

```
$ ./gps-sdr-sim -e brdc3540.14n -l
    30.286502,120.032669,100 -b 16 # For BladeRF
$ ./gps-sdr-sim -e brdc3540.14n -l
    30.286502,120.032669,100  # For HackRF
```

Listing 3. Generate a GPS Baseband Signal Samples for a Static Location at December 20th, 2014

After baseband signal samples are generated, we can transmit them through antenna of SDR platform.

*3) HackRF Transmittion:* We can use the command line below to transmit samples using HackRF, at 1575.42MHz, repeatedly.

```
$ hackrf_transfer -t gpssim.bin -f 1575420000 -s
    2600000 -a 1 -x 0 -R
```

Listing 4. Loop Transmitting the Baseband Samples Using HackRF

Where:

- -t filename, Transmit data from file.
- -f freq_hz, Frequency in Hz.
- -s sample_rate, Sample rate in Hz.
- -a amp_enable, RX/TX RF amplifier 1=Enable, 0=Disable.
- -x gain_db, TX VGA (IF) gain, 0-47dB, 1dB steps.
- -R, Repeat TX mode.

The ‘-R’ parameter is contributed by us to HackRF project to support repeat transmit mode. It’s very handy because ‘gps-sdr-sim’ only generates a sample of three minutes, while GPS receivers probably don’t have enough time to complete the first GPS capture process in just three minutes.

*4) BladeRF Transmittion:* We can use the command line below to transmit samples using BladeRF, at 1575.42MHz.

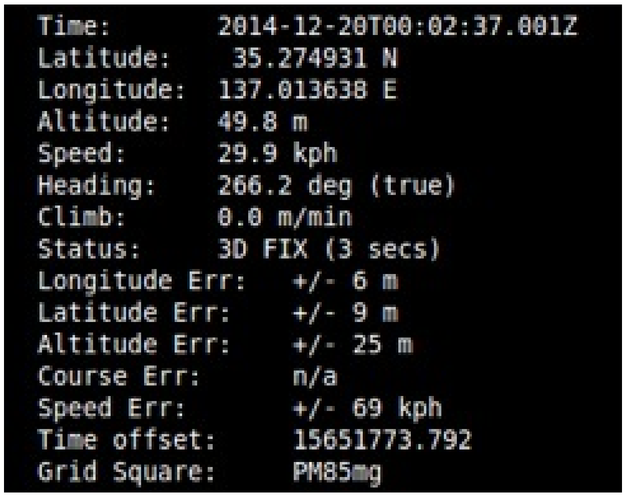
```
bladeRF-cli -s bladerf.script
```

Listing 5. BladeRF Transmit Command

Fig. 3 shows frequency spectrums of GPS samples transmitted by BladeRF.

## C. Experiment Results

| Item | Value |
|---|---|
| Hardware | BladeRF or HackRF |
| Test device | iPhone 6 and Apple Watch (42mm sports version). |
| Test network | China Mobile LTE, China Unicom LTE |
| Real location | Tsinghua University Main building , Beijing, China |
| Real localtime | $\sim$ 2015-06-20 21:00 (UTC+8) |
| GPS spoofing location | Japan, (35.274931N, 137.013638E) |
| GPS spoofing time | 2014-12-20 00:00:00 (UTC+0) |



```
Time:          2014-12-20T00:02:37.001Z
Latitude:       35.274931 N
Longitude:     137.013638 E
Altitude:      49.8 m
Speed:         29.9 kph
Heading:       266.2 deg (true)
Climb:         0.0 m/min
Status:        3D FIX (3 secs)
Longitude Err:    +/- 6 m
Latitude Err:     +/- 9 m
Altitude Err:     +/- 25 m
Course Err:       n/a
Speed Err:        +/- 69 kph
Time offset:      15651773.792
Grid Square:      PM85mg
```

Fig. 4. GPS Faking Demo: GPS receiver

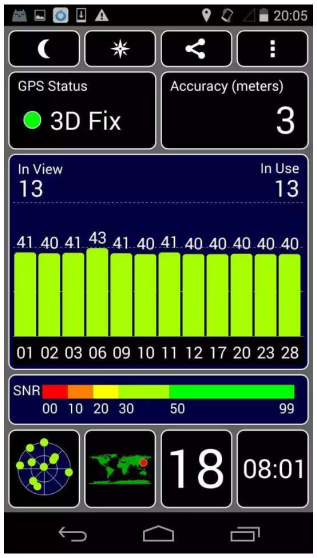


Fig. 5. GPS Faking Demo: Android GPS Test App

1) *iPhone and Android Cellphones:* When GPS spoof starts, the positioning data of iPhone will be cheated, even with WiFi and cellular service switched on. So Apple seems to trust GPS data at quite a high level.

When GPS spoofing is stopped, but without real GPS signal received, date and time wouldn't be fixed immediately according NTP service from WiFi or cellular network.

When we get to the open air to get a strong and stable real GPS signal, the affection won't be fixed for at least 10 minutes even after we switched 'General → Date & Time → Set Automatically' to 'Off' and set back the right time then switched to 'On'.

Fig. 5 shows that a GPS Test app running on Android cellphone, gets a '3D Fix' result.

2) *Time Spoofing on Apple Watch:* The date and time of iPhone and Apple Watch was cheated by GPS signal also, in this case, changed to December 20th, 2014, while the real

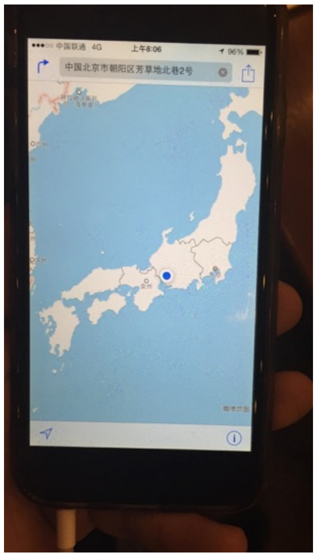


Fig. 6. GPS Faking Demo: iPhone. Real Position: Beijing

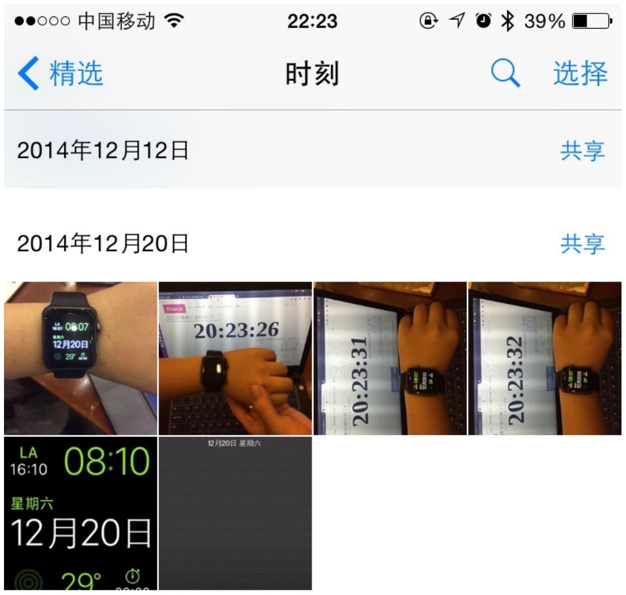


Fig. 7. GPS Faking Demo: Camera timestamp disorder

local date and time is June 20th, 2015. Since Apple Watch is fully synchronized with iPhone, as is shown in Fig. 9.

Fig. 7 shows that photos or screenshot taken with iPhone at this time, will be inserted into photo stream with a false timestamp location.

3) *LBS Apps: Uber, Didi, and Wechat:* Nowadays, more and more mobile apps use positioning data to bring more suitable service to users. But it seems that those apps trust positioning data totally. For example, taxi calling apps like Uber and Didi can be easily cheated using this GPS spoofing method, as are shown in Fig. 10.

Fig. 11 shows that location based apps like WeChat, will get a cheated location when posting a photo with geographic labels.

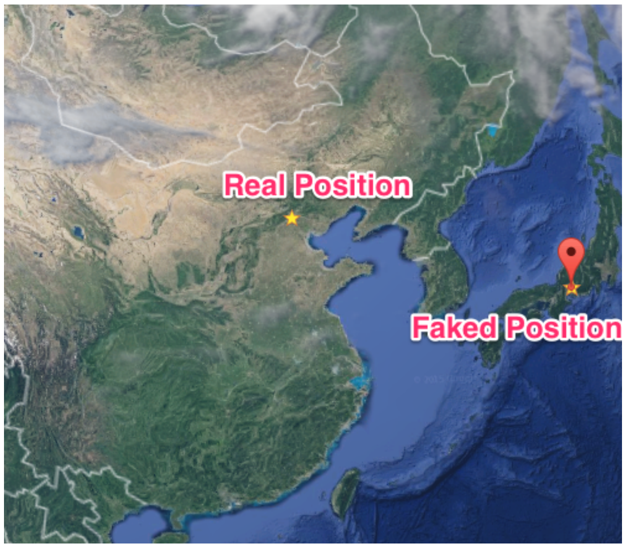


Fig. 8. GPS Faking Demo: Beijing Japan

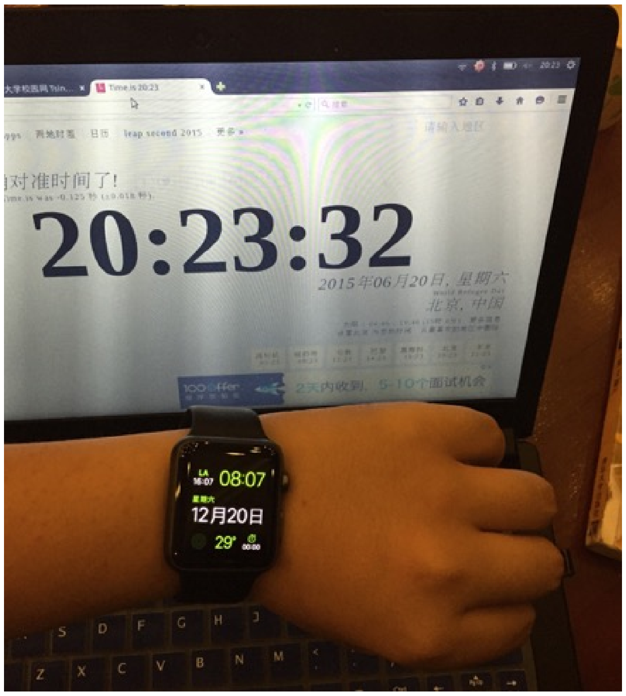


Fig. 9. GPS Faking Demo: Apple Watch. Real Date: 2015-06-20

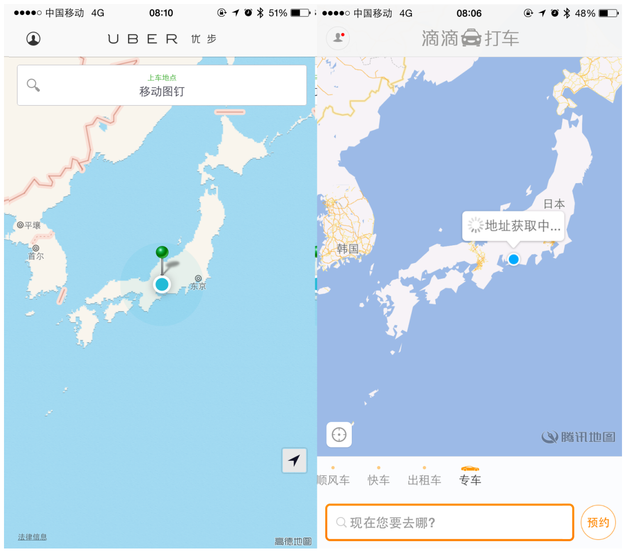


Fig. 10. GPS Faking Demo: Uber and Didi Taxi

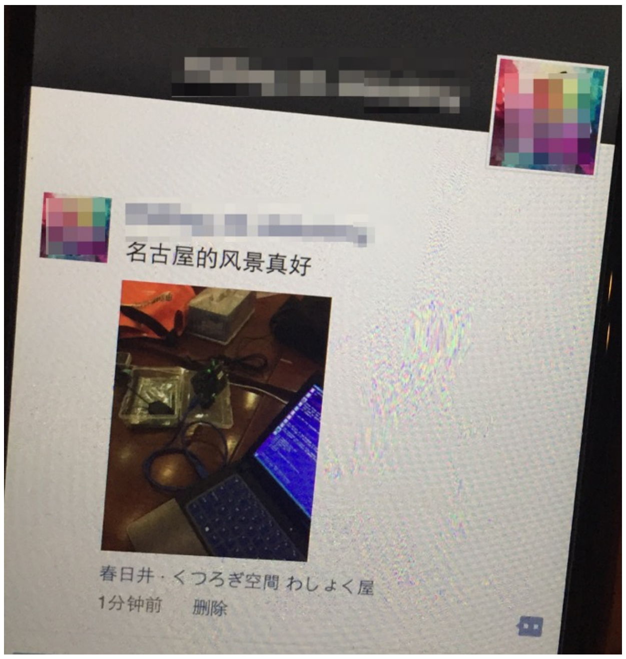


Fig. 11. GPS Faking Demo: Wechat App

*4) Furthermore: Base station frequency offset:* Later, we found out that this GPS spoofing method can also interfere the sync signal of cellular network base stations. Most base station in cellular networks rely on PPS(Pulse Per Second) signal which derived from GPS signal to calibrate frequency offset. We transmit fake GPS signal which contains a high-error PPS signal, then the LTE base station automatically sync with this PPS signal, and the whole LTE network's frequency error increased from 5Hz to ∼11000Hz.

*5) Furthermore: NTP Service:* NTP service provides time synchronization service through Internet connection. Most of NTP services use GPS time as upstream synchronization source.

If NTP service is affected by attackers, the impact would be enormous.

## III. WiFi Based Location Spoofing

### *A. Principle of WiFi Assisted Positioning*

Since GPS positioning won't work indoors, positioning service providers such as Apple Maps, Google Maps, Baidu Maps often use WiFi signal to help users to get a better positioning performance.

The principle is simple. The wireless chipset of cellphone is able to provide site survey information against WiFi hotspot surrounding. The key information to help positioning is SSID and BSSID. SSID (Service Set IDentification) is the display name of WiFi hotspot. BSSID (Basic Service Set Identification) is the MAC address of the wireless access point (AP). Positioning service providers collect SSID and BSSID against GPS data into their positioning assist database, sometimes the information collect process is completed in cellphones of end users.

For example, in Apple Location Q&A [13]: "Rather, its maintaining a database of Wi-Fi hotspots and cell towers around your current location, some of which may be located more than one hundred miles away from your iPhone, to help your iPhone rapidly and accurately calculate its location when requested. Calculating a phones location using just GPS satellite data can take up to several minutes. iPhone can reduce this time to just a few seconds by using Wi-Fi hotspot and cell tower data to quickly find GPS satellites, and even triangulate its location using just Wi-Fi hotspot and cell tower data when GPS is not available (such as indoors or in basements). These calculations are performed live on the iPhone using a crowd-sourced database of Wi-Fi hotspot and cell tower data that is generated by tens of millions of iPhones sending the geo-tagged locations of nearby Wi-Fi hotspots and cell towers in an anonymous and encrypted form to Apple. "

So, what about we generate some fake SSIDs and BSSIDs, in order to see whether those positioning service be cheated or not.

The most straight forward idea is to collect those data manually, then buy a lot of wireless routers, setting the BSSID and SSID of each router according to our needs. But it will be quite a hard job to complete. Thus, we need to come up with a more effective idea.

### B. Collect SSID and BSSID

First, we need to collect SSID and BSSID around. For Linux, we have 'iw' utils which can easily collect those IDs. In order to speed up, we wrote a GNU/awk text substitution script 'wifi-mdk3.awk' in Listing 6 to process 'iw' utils output to fit later use.

```
$1 == "BSS" {
    MAC = $2
    wifi[MAC]["enc"] = "Open"
}
$1 == "SSID:" {
    wifi[MAC]["SSID"] = $2
}
$1 == "freq:" {
    wifi[MAC]["freq"] = $NF
}
$1 == "signal:" {
    wifi[MAC]["sig"] = $2 " " $3
}
$1 == "WPA:" {
    wifi[MAC]["enc"] = "WPA"
}
$1 == "WEP:" {
    wifi[MAC]["enc"] = "WEP"
}
END {
    for (BSSID in wifi) {
        printf "%s %s\n",BSSID,wifi[BSSID]["SSID"]
    }
}
```

Listing 6. wifi-mdk3.awk

```
$ sudo iw wlan0 scan |gawk -f wifi-mdk3.awk >
    scan_result.txt
```

Listing 7. Capture Wireless Around

Scan Results:

```
ec:26:ca:38:25:8a metrust
74:1e:93:63:74:b9 STB_IKPG
4c:09:b4:2e:bc:e5 VIDEOPHONE_zwRu
c8:3a:35:3f:2e:e0 www.wboll.com
a8:15:4d:14:a3:88 DYJL
c4:14:3c:f3:5c:4d Baidu_Mobile
4c:09:b4:2e:83:f4 CU_mcSC
5a:c7:16:fa:e2:94 STB_Wa7a
c4:14:3c:fb:58:3c Baidu_Friend
c4:14:3c:e4:a1:dc Baidu_Friend
c4:14:3c:f3:5c:4f Baidu
00:1f:a4:ed:e6:d0 CU_pngE
00:1f:a4:ed:e6:d1 VIDEOPHONE_pngE
00:1f:a4:ed:e6:d2 STB_pngE
00:1f:a4:ed:e6:d3 BACKUP
ec:17:2f:25:ca:4e bjjfsd-VIP
6c:e8:73:fe:01:ee dhjc
f4:ec:38:58:79:b2 ZJDZGC off
ec:26:ca:b9:a5:d2 zkyc168
14:e6:e4:7e:ad:56 lichunfeng
14:75:90:0f:52:10 bjjfsd01
c4:14:3c:fb:58:ac Baidu_Friend
c4:14:3c:f3:5c:4e Baidu_WiFi
42:0f:0e:20:9c:62 xz-test
32:0f:0e:20:9c:62 XZ-gaoceng
10:0f:0e:20:9c:62 XZ-office
12:0f:0e:20:9c:62 XZ-caiwu
c4:14:3c:fb:58:3f Baidu
c4:14:3c:e4:a1:df Baidu
c4:14:3c:fb:58:3d Baidu_Mobile
c4:14:3c:e4:a1:de Baidu_WiFi
c4:14:3c:e4:a1:dd Baidu_Mobile
c4:14:3c:fb:58:3e Baidu_WiFi
c4:14:3c:fb:58:ad Baidu_Mobile
c4:14:3c:f3:5c:4c Baidu_Friend
c4:14:3c:fb:58:af Baidu
ec:26:ca:6c:09:17 TP_820_5G
14:75:90:2a:b8:3a zjyd
72:c7:16:fc:86:07 STB_E2B9
72:c7:16:fc:86:06 VIDEOPHONE_E2B9
72:c7:16:fc:86:04 BACKUP
b8:c7:16:fc:86:05 CU_E2B9
c0:a0:bb:49:c8:04 martin
00:25:86:a7:b5:82 etsee
80:89:17:b2:dc:d2 OT
14:75:90:31:34:ee hzcs
b8:62:1f:51:84:54 ciscosb1
14:75:90:35:43:0b sdtp
d4:ee:07:10:69:b4 wechat.wboll.com
c8:3a:35:21:f2:b0 Tenda_21F2B0
8e:be:be:2a:7f:f7 Xiaomi_Hello_PZS7
8c:be:be:2a:7f:f5 YF.007
bc:d1:77:2c:96:1a Acoustic
14:75:90:2a:b8:3b zjyd
78:a1:06:54:2a:1e 007
```

Listing 8. scan_result.txt

For Mac OS X:

```
$ cd /System/Library/PrivateFrameworks/Apple80211.
    framework/Versions/Current/Resources/
$ ./airport -s |grep -v unicast |awk '{ print $2 " "
     $1; }'> /tmp/scan_result.txt
```

Listing 9. Scan on OS X

### C. WiFi Spoofing

To complete this attack, we only need a linux laptop with a wireless card.

First, we need to install MDK3, which is a proof-of-concept tool to exploit common IEEE 802.11 protocol weaknesses.

```
$ wget ftp://ftp.hu.debian.org/pub/linux/
    distributions/gentoo/distfiles/mdk3-v6.tar.bz2
$ tar jxvf mdk3-v6.tar.bz2
```

Then change the following line in Makefile in order to make MDK3 compile successfully.

```
# Change this line
LINKFLAGS = -lpthread
# to the following line:
LINKFLAGS = -pthread
```

Listing 10. MDK3 Makefile Patch

In order to generate fake SSID beacons, we need to set the wireless card to monitor mode. There are two way to do this, one is using 'aircrack-ng' package, another is using 'iwconfig'.

```
$ sudo apt-get install aircrack-ng
$ sudo killall wpa-supplicant
$ sudo service stop network-manager
$ sudo airmon-ng start wlan0
$ sudo mdk3 wlan0-mon b -v scan_result.txt
```

Listing 11. SSID beacons flooding using mdk3 and airmon-ng

Or, use iwconfig instead:

```
$ nmcli dev disconnect iface wlan0
$ sudo ifconfig wlan0 down
$ sudo iwconfig wlan0 mode monitor
$ sudo mdk3 wlan0 b -v scan_result.txt
```

Listing 12. SSID beacons flooding using mdk3 and iwconfig

```
$ sudo ./mdk3 --help b
b    - Beacon Flood Mode
         Sends beacon frames to show fake APs at
    clients.
         This can sometimes crash network scanners
    and even drivers!
         OPTIONS:
      [...]
         -v <filename>
             Read MACs and SSIDs from file. See
    example file!
      [...]
```

Listing 13. Usage of MDK3 -b mode

### D. Experiment Results

Soon after we start mdk3 program, a lot of fake SSIDs can be scanned using our cellphone, but those SSIDs cannot be attached successfully, which is as we expected. Then we opened Baidu Map. After a while, the positioning is affected successfully as shown in Fig. 12. Faked position and real position of this case are shown in Fig. 13.

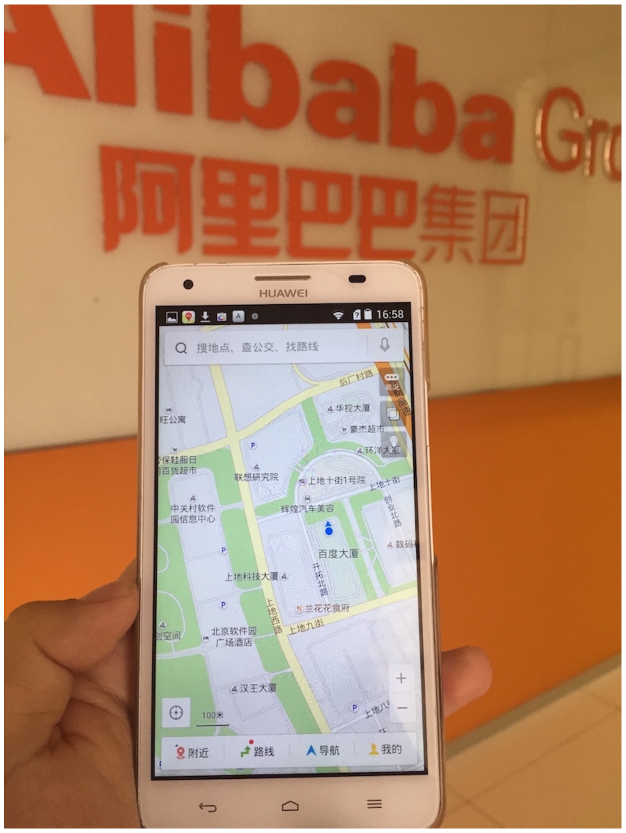


Fig. 12. WiFi Position Faking Demo: Beijing

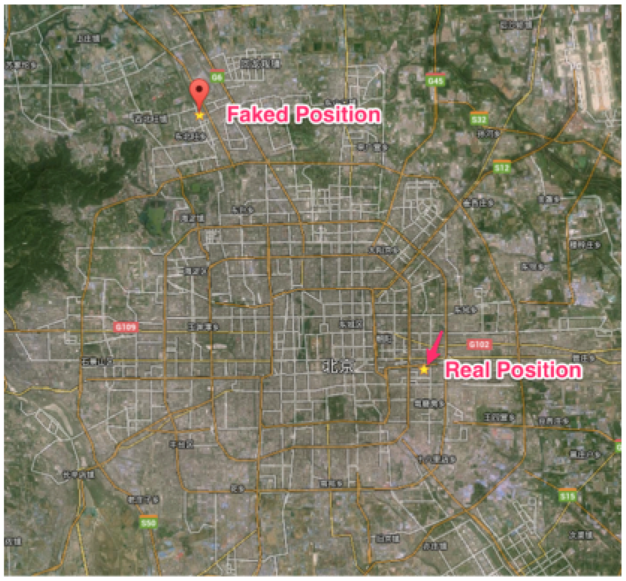


Fig. 13. WiFi Position Faking Demo: Beijing

## IV. Advices

We give some suggestions on positioning security during development as below:

1) Add a position and date time check based on continuous principle. Position and time hopping should be verified or prompted to users in mobile devices.
2) Add a separate clocking hardware module within Apple Watch.
3) Decrease the cache time from GPS positioning signal. According to our observation, the cache time of GPS position data should be decreased, since it's not appropriate for a faked GPS position to be cached for a long period.
4) Add a manually refresh GPS cache function. It's necessary to provide a forced refresh function for position data when user find out that position data is suspicious.
5) Add a high priority time sync service, based on NTP over SSL. Since internet connection is more reliable for mobile devices, an authoritative NTP time synchronization over a safe channel as SSL should be lifted to a higher priority in mobile system.
6) GPS signal strength detect. Fake GPS signals are often much stronger and much more uniform than real signal, as is shown in Fig. 5. Abnormal signal strength change could be used as an detect identification.
7) WiFi positioning data provider should do cross verification for their database in order to exclude fake samples.

## V. Conclusion

In this paper, we have introduced two methods of positioning spoofing. Technical details have already been submitted to Apple Product Security Team as well.

It is important for developers and vendors to examine position and time data more closely and more frequently and effectively. As a basic security principle, any user input shouldn't be trusted completely.

## Acknowledgment

The authors would like to thank Dr. Yang Bo from China Academy of Telecommunication Research (CATR) for his work on frequency offset interference of basestation.

The authors would also like to thank Professor Mingquan LU, we learned a lot from his courses on positioning systems.

The authors would also like to thank TUNA (Tsinghua University TUNA Association) for the help on experiments.

## References

[1] Dong L. IF GPS signal simulator development and verification[M]. National Library of Canada= Bibliothque nationale du Canada, 2005.
[2] Akos, D. M. (1997), A Software Radio Approach To Global Navigation Satellite System Receiver Design, Dissertation, Ohio University.
[3] Kaplan, E. D. (1996), Understanding GPS, Principles and Applications, Boston: Artech House, Inc.
[4] https://play.google.com/store/apps/details?id=com.chartcross.gpstest
[5] http://www.pseudocode.info/post/50127404555/beacons-beacons-everywhere-using-mdk3-for-ssid
[6] https://github.com/osqzss/gps-sdr-sim
[7] https://en.wikipedia.org/wiki/Global_Positioning_System
[8] https://en.wikipedia.org/wiki/GPS_signals
[9] https://en.wikipedia.org/wiki/RINEX
[10] http://cddis.gsfc.nasa.gov/Data_and_Derived_Products/GNSS/broadcast_ephemeris_data.html
[11] BladeRF: http://nuand.com
[12] HackRF: http://github.com/mossmann/hackrf
[13] https://www.apple.com/pr/library/2011/04/27Apple-Q-A-on-Location-Data.html

## Authors

- *Kang Wang* is a security specialist of the mobile security division within the Alibaba Corporation. He focuses on security issue of new technology. He is a contributor of Linux Kernel (TDD-LTE USB Dongle support) as well as a co-founder of the Tsinghua University Network Administrators (http://tuna.tsinghua.edu.cn).
- *Shuhua Chen* is the director of the mobile security division within the Alibaba Corporation. He focuses on finding new technology and new business model to help the industry solve security problems easily.
- *Aimin Pan* is the chief architect of the mobile security division within the Alibaba Corporation. He has written and translated many books, including "Understanding the Windows Kernel"(Chinese edition, 2010) and "COM Principles and Applications"(Chinese edition, 1999). Before joining Alibaba, he worked at Peking University (Beijing), Microsoft Research Asia, and Shanda Innovations. Aimin has published more than 30 academic papers, filed 10 USA patents. In recent years, his research focuses on mobile operating systems and security.