DEMO: Far Away and Yet Nearby - a Framework for Practical Distance Fraud on Proximity Services for Mobile Devices

[Extended Abstract]

Tobias Schultes, Markus Grau, Daniel Steinmetzer, and Matthias Hollick
Secure Mobile Networking Lab, TU Darmstadt, Germany
{tschultes, mgrau, dsteinmetzer, mhollick}@seemoo.tu-darmstadt.de

ABSTRACT

Proximity services are widely used in mobile applications for fast and easy data transfer and control of various systems within a defined range. Authorization is achieved by proximity detection mechanisms that surrogate extensive pairing processes. In this work, we present our Nearby Distance Fraud Framework (NeDiFF) to investigate distance fraud on various proximity services. NeDiFF cheats on proximity checks in services as Google Nearby Messages, Chromecast guest mode and Android device location. Our results emphasize that proximity services currently used for mobile devices are prone to relay attacks and should not be used in security-sensitive applications.

Keywords

Android; Chromecast; Distance Fraud; Google Nearby Messages; Location Spoofing; Proximity Detection; Relay Attack

1. INTRODUCTION

Proximity services for mobile devices are used to identify possible communication partners in a defined range without extensive pairing processes. Based on proximity detection, mobile devices are authorized to exchange messages, control systems or access location dependent information. Such systems provide great advantages, but often are vulnerable to distance frauds [2].

In this work, we investigate the necessary effort to successfully cheat on common proximity services for mobile devices. We practically investigate the feasibility of forging proximity detection in three popular application scenarios. We consider exemplary services that allow (1) message and data exchange with devices in proximity, (2) control of entertainment systems, and (3) access to location dependent information. Our Nearby Distance Fraud Framework (NeDiFF) demonstrates the practicality of distance fraud with common resources: our tool-set only consists of laptop, Android devices and custom software. The results show that neither services guarantees proximity: all investigated scenarios are prone to distance frauds.

The remainder of this paper is structured as follows. In Section 2, we describe the proximity services considered in this work. We propose NeDiFF and evaluate the feasibility of distance fraud in Section 3. We discuss our results in Section 4 and finally conclude this work in Section 5.

WiSec'16 July 18-22, 2016, Darmstadt, Germany

ACM ISBN 978-1-4503-4270-4/16/07.

DOI: http://dx.doi.org/10.1145/2939918.2942416

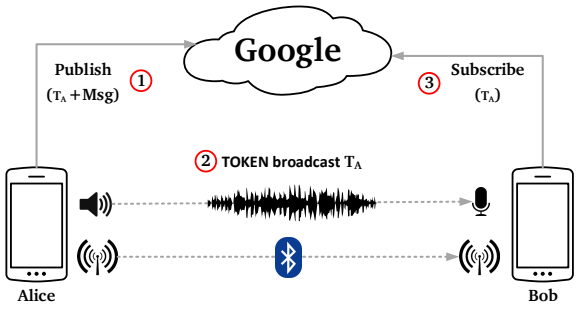

Figure 1: Nearby Messages operation principle

2. PROXIMITY SERVICES

To address various application scenarios, we investigate distance fraud on different popular proximity services. In particular, we consider (1) Google Nearby Messages, (2) Chromecast guest mode, and (3) Android device location. Each of these services is described in the following.

2.1 Google Nearby Messages

Google Nearby Messages [1] is a publish/subscribe API for devices in proximity, which is part of Google APIs for Android. Figure 1 shows the operation principle with Alice publishing a message ①. The proximity detection is achieved by local transmissions of tokens over Bluetooth or ultrasonic audio ②. Bob receives Alice's token and sends it to Google to perform the subscription ③. The messages itself are always delivered through the Google cloud.

2.2 Chromecast Guest Mode

Chromecast is a media streaming device from Google, which can be plugged into TV to e.g. stream movies from a smartphone. In guest mode, the streaming device must be in the same room but not necessarily connected to the same network. Location checks are performed with a com-

bination of WiFi and ultrasound signals. The smartphone receives WiFi beacons from the Chromecast and its associated access point to identify Chromecasts in proximity. Further, the smartphone must receive an ultrasonic audio token transmitted over the TV to approve that both devices are in the same room. If the latter check fails, a backup solution is to enter a PIN on the smartphone.

2.3 Android Device Location

A different approach is the usage of absolute positions for proximity estimation as typically performed on Android devices. A lot of mobile applications like Google Maps or Ingress[1] utilize this to determine the distance to points of interest. On Android, the position can be resolved with GPS and also by WiFi. The latter uses databases of MAC addresses, SSIDs and GPS positions of known access points. Scanning all WiFi networks in range allows to query respective positions.

3. DISTANCE FRAUD FRAMEWORK

In this section, we describe NeDiFF, our distance fraud framework and explain the identified methods that allow to perform distance fraud on the aforementioned proximity services. This incorporates token relaying on the audio and Bluetooth channel in Google Nearby Messages, location spoofing for Chromecast guest mode and Android device location as described in the following.

3.1 Google Nearby Messages

In Google Nearby Messages, we assume one device to share a message within a bounded area. To access this message from outside this area, the token can be recorded in range, transfered to a remote location and replayed again. This allows other devices at the remote location to pass the proximity check and retrieve the message from Google. This is a classical relay or wormhole attack (similar to [3]) that in Google Nearby Messages can performed on two mediums: the audio and the Bluetooth channel as described in the following.

Google Nearby Messages uses inaudible ultrasonic frequencies for token transmission on the audio channel. The sound can be recorded with a common microphone (as available in every phone) and transmitted to a remote place. There, the sound is replayed again using a common smartphone speaker. Our evaluations did not indicate any significant quality losses in this amplify-and-forward approach. Remote devices were able to retrieve the shared message.

On the Bluetooth channel, the token is published by setting its value as device name. The relay just needs to scan for Bluetooth devices in range and forwards their names. This follows the decode-and-forward approach as the remote end simply changes the Bluetooth interface name. Remote devices cannot distinguish, whether a certain device is the original or relayed one. As the transmitted amount of data is lower as in the audio relay, this approach performs faster, comes to the same result and was therefore used in NeDiFF.

3.2 Chromecast Guest Mode

Our framework supports remote devices to gain access to a Chromecast in guest mode without having physical access. However, to achieve this, several steps are necessary. As a prerequisite it has to be known to which WiFi access point the Chromecast is connected to. This knowledge can for example be obtained by eavesdropping WiFi communication near the site of the Chromecast. MAC address and SSID of this access point have to be replayed at the remote location. Chromecast compatible applications attempt to connect as soon as a WiFi beacon with a Chromecast MAC is received. To accomplish the pairing, an audio token or four-digit PIN is required. The first can be achieved similar to the audio relay method in the Google Nearby Messages, the later can be guessed. Brute-forcing four-digits is assumed to be feasible with low effort but out-of-scope of this work. As Chromecasts do not punish failed connection attempts, this task becomes non time-critical. Using this approach, adversaries become able to stream arbitrary content to any Chromecast, even when the TV is in standby.

3.3 Android Device Location

The authors of [4] showed that Skyhook, a WLAN-based positioning system is prone to location spoofing. We used their approach against the positioning service integrated in Android and revealed that it is not protected against this kind of attacks, too. To perform location spoofing on such services, a WiFi fingerprint of the desired position is needed. The single access points can be cloned at a remote location by impersonating the particular MAC addresses and SSIDs. We integrate the scanning and relaying of access points in NeDiFF. Hence, we can pretend devices to be at completely different locations where we only scanned once for available networks. As all received access points are taken into account for position resolving, this method has higher success rates at remote locations with low WiFi density.

4. DISCUSSION

Our investigations with NeDiFF have shown that all three proximity services can be fooled by low-cost relay attacks. Unauthorized users at remote locations might stream videos onto any Chromecast connected TV with guest mode enabled or access Google Nearby Messages based application data (e.g. voting for tracks in Edjing playlists). They can also trick devices with spoofed locations to access location dependent information without residing at the right position. While cheating on the approach used in Google Nearby Messages requires a device within the intended range permanently, WiFi access points provide static information with long validity, which have to be obtained only once. Additionally WiFi beacons can easily be cloned and therefore are no proper method for proximity detection.

5. CONCLUSION

Distance fraud is a known threat on localization systems. Despite the fact that no suitable solutions exist, billions of devices makes use of proximity services for mobile devices. The advantages of delivering location aware content often dissolve the risks of location spoofing. However, the threat is no longer theoretical. We aim to increase awareness of this risk and propose NeDiFF, a low-cost framework to practically perform distance fraud on different proximity services. Our results show that neither service withstands distance fraud and should not be used in security-sensitive applications.

[1] https://www.ingress.com

6. ACKNOWLEDGMENT

This work has been funded by the DFG within CROSSING, the BMBF and the State of Hesse within CRISP-DA, and the Hessian LOEWE excellence initiative within CASED.

7. REFERENCES

[1] Google Nearby Messages API. https://developers.google.com/nearby/messages/overview. Accessed: 2016-05-13.

[2] J. Clulow, G. P. Hancke, M. G. Kuhn, and T. Moore. So near and yet so far: Distance-bounding attacks in wireless networks. In *Security and Privacy in Ad-hoc and Sensor Networks*, pages 83–97. Springer, 2006.

[3] M. Maass, U. Müller, T. Schons, D. Wegemer, and M. Schulz. NFCGate: an NFC relay application for Android. In *Proceedings of the 8th ACM Conference on Security & Privacy in Wireless and Mobile Networks*, page 27. ACM, 2015.

[4] N. O. Tippenhauer, K. B. Rasmussen, C. Pöpper, and S. Čapkun. Attacks on public WLAN-based positioning systems. In *Proceedings of the 7th international conference on Mobile systems, applications, and services*, pages 29–40. ACM, 2009.

8. APPENDIX: DEMONSTRATION

During the demonstration, we show distance fraud on the three proximity services: (1) Google Nearby Messages, (2) Chromecast guest mode, and (3) Android device location. In particular, we extend the operation range in Google Nearby Messages and Chromecast guest mode and spoof location of Android devices. In the following, we first describe the basic environmental setup for the demonstration and then discuss the individual distance frauds on each of the proximity services.

8.1 Environmental Setup

To demonstrate distance fraud, we need to spread our setup over different locations. We place devices at (1) the conference booth, (2) a shielded area at the conference booth and (3) a remote location far away from the conference location. Doing so, we become able to show that the operation range of proximity services can be extended to locations different than the intended area. At the conference booth, we place devices for interaction with conference participants. The shielded area is used for smartphones at the conference booth to emulate spatial distance by blocking the WiFi and Bluetooth frequency bands. At the remote location, we place devices that require no user interaction and use a webcam to monitor the outcome. All together, we use four Android devices, a laptop, a Chromecast and a common TV.

8.2 Google Nearby Messages

The first demonstration shows how to fool the proximity detection of Google Nearby Messages with NeDiFF. Therefore, a setup of four Android devices is necessary, as illustrated in Figure 2. Initially, a simple Android application called Nearby Devices is used to show the normal publish/subscribe behavior of Google Nearby Messages. At this point the proximity check for the shielded area will fail and no messages be delivered. Unshielded devices indeed receive the messages. The developed Android application NeDiFF extends the range by relaying the Bluetooth channel into the shielded area. Doing so, the proximity check passes and the remote device becomes able to subscribe and receives the messages.

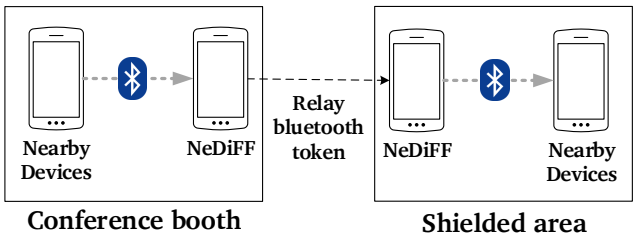

Figure 2: Google Nearby Messages demonstration

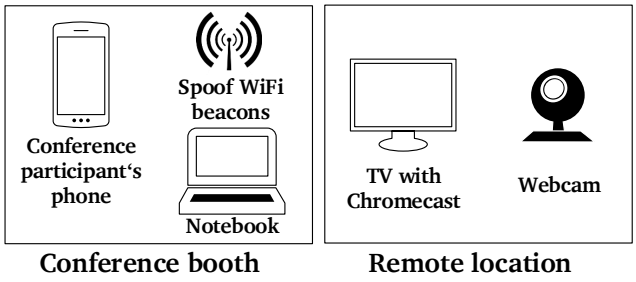

Figure 3: Chromecast demonstration

8.3 Chromecast Guest Mode

The second demonstration extends the operation range and fools Google Chromecast's guest mode built-in proximity check. As shown in Figure 3 the conference participant is directly involved with his own Android smartphone at the conference booth. He can use an application which implements the Google Cast API (e.g. Youtube) to stream content to the remote location. At the remote location, a Chromecast is connected to a TV and observed by an installed webcam. The participant can first try to establish a connection to the remote Chromecast, which initially fails. After unsuccessful tries, a laptop is started to broadcast previously captured WiFi beacons from Chromecast's location. Doing so, the conference participant becomes able to connect and stream arbitrary content to the remote Chromecast.

8.4 Android device location

The last demonstration shows how easy Android device locations can be spoofed. This involves conference participants' Android devices. First, the phone's GPS positioning is turned off to rely on WiFi localization only. Then, the current position of the device is checked with e.g. Google Maps. We use a setup similar to the left part of Figure 3. Both devices are located at the conference booth, but placed in a shielded area to suppress beacons of local WiFi networks. The phone will receive the spoofed beacons broadcasted by the laptop and update its position on Google Maps subsequently. In advance, an exemplary list of fingerprints to fake was captured with NeDiFF. These contain the MAC addresses and SSIDs of all access points at a specific location.