POSTER: Android + Open Wi-Fis = Broken SSL?

Sascha Fahl
Dept. of Computer Science
Leibniz University Hannover

1. INTRODUCTION

In previous work [1] we demonstrated severe problems with the way Android applications use SSL. We performed an in-depth study of 13,500 Android apps and discovered that a large number of apps did not use SSL correctly and thus, were vulnerable to Man-In-The-Middle attacks.

To make these threats a reality, an attacker needs to execute an active man-in-the-middle attack (MITMA). While MITMAs are a threat against desktop systems as well, MITMAs against mobile devices are easier to mount, since the use of mobile devices frequently occurs in changing and untrusted environments. In this work, we evaluate the severity of the threat of MITMAs against Android devices which use public Wi-Fis and show how the problems with SSL and the CA infrastructure not only do not protect from MITMAs but can actually facilitate them. While the use of open access points and the evil twin attack [2] are already well known threats against open Wi-Fis, we also show how attackers can even bypass apps that implement secure SSL certificate verification.

We conducted an initial Amazon MTurk based study of our attack for Android users which can be used to set up future MITMAs even when SSL is implemented correctly.

2. ANDROID, OPEN WI-FIS AND MITMAS

In our Android analysis[1] we concentrated on the issues app developers have with applying SSL correctly. However, the complexity of SSL and the underlying CA approach also creates serious problems for end users. We discovered one very critical problem in combination with captive portals of open Wi-Fis and Android. Open Wi-Fis are a known problem area concerning evil twins, MITMAs and the harvesting of unencrypted information sent via these unsecured networks [2]. Usually, public hotspots are "authenticated" based only on their SSID. Since SSIDs can be chosen at will, this presents no protection at all. While the popular press has covered this issue, people still fall for this simple trick quite easily.SSL was designed to protect from exactly this kind of scenario. Any site a user surfs to using HTTPS is protected, even in the face of a MITMA, as long as the attacker was not able to obtain a valid certificate for the server in question. While the compromises of DigiNotar and Comodo show that this is actually a possibility, it is not a trivial attack. However, there is an inherent weakness in the CA-based SSL model that can be used in combination with a social engineering attack to circumvent SSL with very little effort.

Operating systems for desktop computers and mobile devices all have a list of trusted root CAs (usually between 100 and 200 CAs) that can be extended by the user. On Android this feature can for instance be used to install a particular company-specific root certificate authority.

Based on this, we developed a social engineering attack that is critical in general, but threatens users of public hotspots in particular:

2.0.1 The Trust Injection Attack

We present a novel attack against Android, exploiting captive portals of public Wi-Fi hotspots. To connect to such a Wi-Fi network, users select the SSID of the network and are then presented with a captive portal website on which they have to accept the terms of use and then click a button to go online. In our attack, the operator of a malicious Wi-Fi hotspot presents a user with a captive portal website that promises not only free but also more secure Wi-Fi. To "increase" security, the attacker asks the user to add a *"security certificate"*, which actually is a root CA certificate for which the attacker controls the private key. The attacker forces the user to add the certificate before granting access to the Internet. To add this certificate, the user has to click on a button labeled for instance *"Add Certificate and Connect"* in the captive portal. Figure 1 shows what this could look like for an attacker spoofing the Starbucks free Wi-Fi. Next, the user is asked to enter a name for the certificate and to click ok (cf. Figure 2). If the certificate was not installed, the user is redirected to the captive portal and denied access to the Internet.

Users that fall for this attack have just installed a new root CA certificate in their central keychain that can be exploited to sign arbitrary certificates and are now using the attacker's Wi-Fi hotspot to go online. In this case, the attacker, apart from just mounting conventional active Man-In-The-Middle attacks, is also capable of generating SSL certificates for all hosts the victim is connecting to on the fly. The attacker signs these certificates with the corresponding private key of the root CA certificate the victim just installed into the device's global key chain. Hence, the attacker can effectively

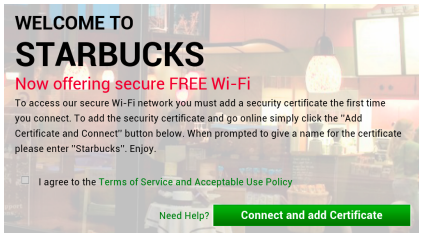

Figure 1: Our modified Starbucks captive portal promises "Secure FREE Wi-Fi"

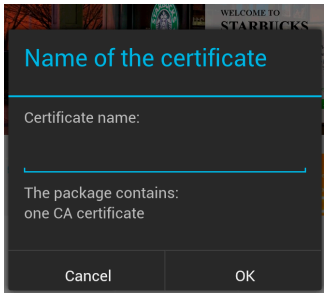

Figure 2: After clicking the "Connect and Add Certificate" button the Wi-Fi user is asked to enter a name for the certificate and click OK.

generate SSL certificates for all SSL connections. As long as an app does not implement SSL pinning – something very few apps appear to do (cf. [1]) – the conventional SSL certificate validation procedure does not protect against this attack.

Since the installation dialog for root CA certificates on Android devices (cf. Figure 2) does not warn the user against possible security threats stemming from the installation of a root CA certificate, nor does it even warn that the certificate will be permanently installed into the global keychain and stored on the device, we think that Android users are particularly vulnerable for this type of attack.

2.0.2 *User Study*

To evaluate this, we conducted an initial study using Amazon's MTurk service. We created a HIT saying that we are conducting a usability study for a new login procedure for free Wi-Fi provided in Starbucks coffee shops. Workers had to be Android users, they must have used the Starbucks Wi-Fi before, and they had to use their Android smartphone to work on the task. We compensated each complete result with $ 1.50.

Design.

Participants were presented with our modified captive portal website, simulating the Trust-Injection attack introduced above (cf. Figure 1). After they clicked on "Connect and Add Certificate", they were redirected to a mock installation dialog for a new root CA certificate (cf. Figure 2). Since the dialogs only mimicked the true CA install process, we did not install anything on the users' phones.

Participants were asked to enter a name for the certificate and click "ok" before they were forwarded to a questionnaire. In the first part of the questionnaire, we asked questions on how participants perceived the usability, security and possible privacy threats of the new captive portal mechanism in comparison to the conventional solution and asked if they would use this login mechanism. The usability was collected using the SUS usability scale. We were interested in these questions to see how users judge the usability of our attack, since accepting a new technology is strongly connected to its usability in addition to perceived security. Finally, we collected demographics and asked for self-reported tech-ratings as well as the Westin-index questions.

128 participants successfully completed the HIT of which 62.5 % were male and 37.5 % were female with an average age of 29.6 years ($sd = 7.95$) and a very high average self-reported tech-rating of 1.57 ($sd = .78$) on a scale from 1 "high expertise" to 5. Participants indicated their occupation as student (22.7 %), part-time employee (10.2 %), full-time employee (45.3 %), self-employed (12.5 %) and Homemaker or unemployed (9.4 %). According to the Westin-index, most of our participants belonged to the privacy pragmatist category (69.5 %), 22.7 % were fundamentalists that value their privacy highly, and 7.8 % are privacy unconcerned. 91.4 % of the respondents use a Starbucks Wi-Fi at least once a month and 39.1 % at least weekly.

Almost three quarters of our participants (73.4 %) accepted the dialog informing them that they were about to install a new root CA by clicking "OK". Of these, 76.6 % believed that installing an additional root CA increases their privacy protection, while 21.3 % believed there was no change and only 2.1 % suspected their privacy to be at risk after installing a new root CA. The majority of participants that clicked "Cancel" believed that installing a new root CA had no effect on their privacy (64.7 %).

Participants also felt that the usability of our trusted root CA injection attack was good: those who installed the root CA provided an average SUS usability score of 76.97 ($sd = 1.78$) with 100 being perfect usability and those who did not install the root CA scored slightly less, at 60.15 ($sd = 4.17$). The differences in usability and increase in security were statistically significant (Mann-Whitney U, $Z = -3.68$ and $Z = -6.528$, $p < .001$).

Although our MTurk study only was a simulation of the Trust-Injection attack, we believe our results indicate that Android users of open Wi-Fi hotspots are particularly threatened. In future work we plan to investigate the potential threat for other platforms and conduct controlled experiments to gain better understanding of the problem. We are currently conducting usability studies on how to design user interfaces for installing root CA certificates on Android devices, since we think the current interface does not sufficiently protect their users from Trust-Injection attacks.

3. REFERENCES

[1] S. Fahl, M. Harbach, M. Muders, T. Smith, L. Baumgärtner, and B. Freisleben. Why Eve and Mallory Love Android: An Analysis of Android SSL (In)Security. In *Proceedings of the 19th ACM Conference on Computer and Communications Security*. ACM Press, Oct. 2012.

[2] Y. Song, C. Yang, and G. Gu. Who is Peeping at Your Passwords at Starbucks? – To Catch An Evil Twin Access Point. In *IEEE/IFIP International Conference on Dependable Systems and Networks*, pages 323–332, 2010.