A Tangled Mass: The Android Root Certificate Stores

Narseo Vallina-Rodriguez
ICSI
Berkeley, CA, USA
narseo@icsi.berkeley.edu

Johanna Amann
ICSI
Berkeley, CA, USA
johanna@icir.org

Christian Kreibich
ICSI / Lastline
Berkeley, CA, USA
christian@icir.org

Nicholas Weaver
ICSI / UC Berkeley
Berkeley, CA, USA
nweaver@icsi.berkeley.edu

Vern Paxson
ICSI / UC Berkeley
Berkeley, CA, USA
vern@icir.org

ABSTRACT

The security of today's Web rests in part on the set of X.509 certificate authorities trusted by each user's browser. Users generally do not themselves configure their browser's *root store* but instead rely upon decisions made by the suppliers of either the browsers or the devices upon which they run. In this work we explore the nature and implications of these trust decisions for Android users. Drawing upon datasets collected by Netalyzr for Android and ICSI's Certificate Notary, we characterize the certificate root store population present in mobile devices in the wild. Motivated by concerns that bloated root stores increase the attack surface of mobile users, we report on the interplay of certificate sets deployed by the device manufacturers, mobile operators, and the Android OS. We identify certificates installed exclusively by apps on rooted devices, thus breaking the audited and supervised root store model, and also discover use of TLS interception via HTTPS proxies employed by a market research company.

Categories and Subject Descriptors

C.2.3 [**Computer-communication networks**]: Network Operations; D.4.6 [**Operating Systems**]: Security and Protection; E.3 [**Data Encryption**]: Public Key Cryptosystems

Keywords

Mobile; TLS; HTTPS; man-in-the-middle; Root Store; Android; public-key infrastructure; X.509; certificates; security; measurement

1. INTRODUCTION

While most mobile device users realize that handset vendors and mobile network operators may customize their devices' firmware, UI, and pre-loaded apps, the effect of these alterations on the device's basic security posture are not widely recognized. The X.509 certificate root store population of a device defines the trust relationships that secure its network communications. Bloated or compromised root stores significantly increase the attack surface of the device, with potentially serious consequences for the user. In this paper, we draw upon on datasets collected by Netalyzr for Android as well as the ICSI Certificate Notary to characterize and evaluate the root stores of thousands of mobile devices in the wild. We make three contributions:

1. Using data from thousands of handsets in active use around the world, we describe vendor and operator-specific additions to the official Android Open Source Project (AOSP) root store populated by Google. Using the ICSI Certificate Notary, we evaluate the extent to which we see these certificates in use in large networks around the world.

2. We identify and describe how third-party apps executed on rooted handsets break the existing model of supervised and audited root stores. Any malicious app can add and remove certificates in the root store without any user awareness, making mobile users more vulnerable to MITM attacks.

3. We report on a case of TLS interception on proxied mobile devices for the purpose of user profiling and marketing, and analyze the mobile apps and domains it affects.

As reported in previous studies [26, 29], removing unnecessary certificates from the root store and tightly controlling certificate additions improves HTTPS security. Any of the certificate authorities (CAs) in the system can lead to violations of the apps' trusted communication channels.

2. BACKGROUND

As a key component of today's Internet security, the Transport Layer Security (TLS) protocol provides endpoint authentication using the X.509 certificate infrastructure. Trusted certificate authorities (CAs) sign server certificates. Clients can verify these certificates using a list of trusted root CA certificates shipping with their browser or operating system. As CAs can sign certificates for *any* site on the Internet, they form one of the weakest links of the global trust hierarchy. When a CA suffers compromise, the attacker (or the entity compelling it) can obtain certificates which enable TLS interception attacks on any target domain [27].

More technically, X.509 certificates contain a public key, a subject string that identifies them, information about the issuer of the certificate, as well as a signature by that issuer.[1] Tampering with the

[1]According to RFC 5280 [23], signature computation involves taking the hash of the names of the subject and issuer, a public key associated with the subject (RSA Key modulus), a validity period, version number and a serial number, signing it with a signature algorithm specified in the certificate and the private key of the certificate.

CoNEXT'14, December 2–5, 2014, Sydney, Australia.

http://dx.doi.org/10.1145/2674005.2675015.

	Android version					
Operating system	4.1	4.2	4.3	4.4	iOS7	Mozilla
No. certificates	139	140	146	150	227	153

Table 1: Number of certificates in different root stores.

certificate will invalidate the signature. Server certificate validation requires following a chain of signatures from the server certificate to a trusted CA certificate contained in the root-store, potentially crossing several intermediate CA certificates.

Before Android 4.0, each app had to maintain its own root store if it wanted to validate certificates. In later versions, Android incorporates a system-wide read-only root store[2] and an API that regulates application access to it. Nevertheless, apps can still maintain their own root store [20, 21].

Table 1 shows the number of root certificates included in Android's official AOSP distribution as supported by Google. Apps automatically consider these certificates secure and trustworthy despite the fact that some of the CAs included, such as Comodo and Türktrust, got compromised in the past [16]. Certificates which do not chain to any of these root certificates can evoke a visual warning message in apps implementing techniques like cert pinning such as Twitter and Facebook apps when encountered [20]. The AOSP root store has increased in size in each consecutive release but currently still weighs in below the corresponding Mozilla and iOS7 root stores. The AOSP root store currently contains one certificate, by *Autoridad de Certificacion Firmaprofesional*, which expired in Oct. 2013. In fact, 117 of AOSP 4.4's 150 certificates also exist in Mozilla's root store. In addition, hardware vendors and mobile operators can (and do) add their own trusted certificates to the root store in the firmware they ship on their own Android handsets.

Any Android user can manually add, disable or delete any certificate in the root store through system settings. While the root store by default only provides read access (i.e., only applications and services with system permissions can modify it) a rooted operating system still allow alterations. Any application with root permissions (either because they are executed on rooted handsets by the users or because they perform root exploits [22]) can add new certificates and delete or modify existing ones. This notably increases the attack surface for such Android users. Despite the fact that a mobile app can protect itself from fraudulently issued certificates using techniques such as certificate pinning and customary X.509 checks, the fact that SSL library developers delegate the responsibility to implement such techniques to application developers [12, 15, 21] means that apps frequently do not employ those checks correctly. Android 4.4 detects and prevents the use of fraudulent Google certificates used in secure SSL/TLS communications. Unfortunately, as opposed to Mozilla's root store, Android does not support specifying trust levels for different CA certificates: they can be used for any operation from TLS server verification to code signing [26].

3. RELATED WORK

To our knowledge, to date there has been no in-depth analysis of the root stores of active Android devices or the root certificates added by hardware vendors. Previous work by Perl et al. [26] compared the root stores of several operating systems to Android, but did not examine differences between devices. They concluded that a large number of certificates can be removed from most root-stores as they are not used for HTTPS traffic. Our analysis confirms this result. Fahl et al. [20] and Georgiev et al. [21] examined security issues of Android applications validating certificates by performing MITM attacks and detailed app-oriented tests (they do not examine the root store of devices). They argued that one of the main problems in mobile security is that users still need to see and interpret the warning messages correctly. Amrutkar et al. [18] measured the display of security indicators in mobile browsers, but they did not evaluate differences in root stores. Huang et al. [24] analyzed MITM attacks on Facebook using forged certificates to intercept encrypted traffic Via embedded Flash they obtained copies of the presented certificates, finding a low number of potential exploit attempts (mainly generated by anti-virus software and corporate-scale content filters). In our work, we instead examine the presumed-trustworthy certificates installed on Android devices.

[2] Android root certificates reside in `/system/etc/security/cacerts/`.

Device model	No. sessions	Manufacturer	No. sessions
Samsung Galaxy SIV	2,762	Samsung	7,709
Samsung Galaxy SIII	2,108	LG	2,908
LG Nexus 4	1,331	ASUS	1,876
LG Nexus 5	1,010	HTC	963
Asus Nexus 7	832	Motorola	837

Table 2: Top 5 mobile devices and manufacturers in our Android dataset.

4. DATASET AND METHODOLOGY

To study the root stores of a large number of providers and handset models, we require a distributed platform enabling the measurement of mobile device information at a global scale. To this end, we use two tools implemented and maintained by ICSI: *(1)* Netalyzr for Android [9, 10] to measure *(i)* the root certificates installed on Android devices, *(ii)* the full trust chain for a collection of popular domains and mobile-services, and *(iii)* a broad spectrum of connectivity aspects; *(2)* the ICSI SSL Notary [13, 17] to *(iv)* validate and classify the observed certificates, *(v)* examine how often Internet services (on any port, not only HTTPS) use the certificates present in Android root stores, and *(vi)* the number of TLS certificates that each root certificate collected by Netalyzr can validate.

4.1 Netalyzr

We collected information about 2.3 million root certificates in 15,970 Netalyzr executions between November 2013 and April 2014. Only 314 root certificates are unique based on the certificate signature. In order to preserve user privacy, we do not collect information that could identify a device or a user uniquely such as the IMEI. We use tuples of recorded WiFi and cellular networks, public IP addresses, handset model, and OS version as a proxy for device identity. Based on the number of unique tuples, we estimate that the Netalyzr session set covers at least 3,835 different handsets and 435 device models. Most of the models belong to a small number of handset manufacturers, as shown in Table 2.

We inspected Google's official Android source code repository to obtain the root certificates maintained by Google. With that information in hand, we grouped the certificates Netalyzr encountered by manufacturer, mobile operator, and handset. We then established certificate identity based on unique fields (RSA key modulus and signature string) and compared each certificate to its equivalent AOSP version root store. Since different Android versions format certificate information differently, we had to inspect the subject and issuer fields manually. If none or more than one value is identified, we crosscheck the signature with Mozilla's root store or in the certificate authority's website to obtain a single value. We analyzed rooted handsets separately from operator and manufacturer root stores to

avoid any bias, as users and third-party apps have permissions to modify the root store. We report on them separately in Section 6.

4.2 The ICSI Certificate Notary

As opposed to related efforts such as Convergence [4] and the EFF's SSL observatory [5], ICSI's Certificate Notary collects certificates passively from live upstream traffic to any port at 8 research and university networks [16], aggregating them into a central database. The certificates collected reflect the activity of about 300K active users, including any traffic originating from mobile devices connected to the network over WiFi. We started the data collection in February 2012. Currently the Notary contains more than 1.9 million unique certificates. Out of these, one million have not expired. The Notary also contains information about more than 66 billion SSL sessions as well as the certificates included in Android, iOS7, and Mozilla official root stores.

We again use the RSA key modulus and the signature of the certificates as collected by Netalyzr to compare them with the Notary's entries for validation. When comparing certificates in different root stores, one needs to note that even though root certificates are not byte-equivalent they can still be "equivalent" if their subject and RSA key modulus are identical (i.e., when they can validate the same child-certificates). In most cases, only the expiration date change.

5. ANDROID ROOT STORES IN THE WILD

The scatter plot shown in Figure 1 details the distribution of Netalyzr devices depending on the number of AOSP certificates (x-axis) and non-AOSP certificates (y-axis, in square-root scale to bring out the differences in lower values) grouped by manufacturer and OS version. The dashed vertical lines indicate the number of certificates present in the official AOSP distribution, as distributed by Google. Despite the fact that most devices have the same number of certificates in their root stores as in their equivalent AOSP distribution, 39% of sessions also have additional certificates. Only 5 handsets were missing some certificates present in their respective AOSP distributions.

The number of additional certificates varies depending on handset manufacturer and OS version. More than 10% of Android 4.1 and 4.2 devices—mainly HTC, Motorola (4.1 and 4.2), LG (4.1 and 4.2) and Samsung 4.4 devices—expand the AOSP certificate set by more than 40 additional certificates. On the other hand, Motorola (4.3 and 4.4 versions), Huawei, Sony, and Asus devices have a similar root store to their equivalent AOSP distribution, with fewer than 10 additions. In the case of some Sony 4.1 devices, we identified a certificate was added which is also present in newer AOSP versions.

5.1 Manufacturer and vendor-specific certificates

This section explains the role played by mobile operators and hardware manufacturers for the 39% of handsets with extended root stores. Figure 2 maps the additional certificates issued per Android version for the most popular manufacturers and mobile operators in our dataset. We omit handset manufacturers and operators with fewer than 10 sessions exhibiting modified root stores. The size of the marker represents the ratio of sessions in which Netalyzr identified a given certificate to the total number of sessions with modified root stores for a given manufacturer or operator. The shape represents whether the Notary identified the certificate as part of iOS7 and Mozilla root stores simultaneously (6.7%), iOS7 root store exclusively (16.2%), Android-specific (37.1%), or if the Notary has

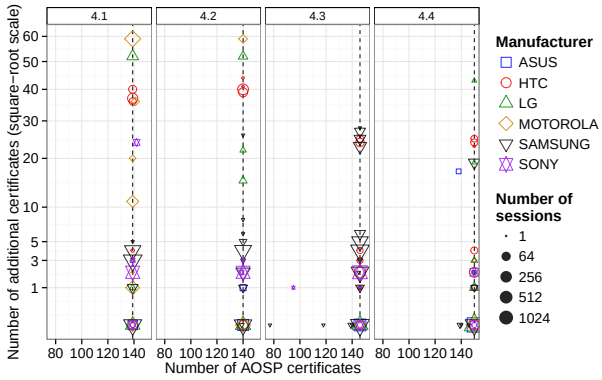

Figure 1: Scatter plot showing the number of devices (size of the symbol in *log*2 scale) with a given number of AOSP certificates (x-axis) and extra root certificates (y-axis, square-root scale) per OS version and manufacturer. The vertical line shows the number of root certificates present in Google's official AOSP distribution.

no record of the certificate at all (40.0%). The latter group contains certificates that may serve uncommon or offline operations such as code signing or firmware updates that the Notary never encounters in network traffic.

Non-AOSP certificates can be issued by well-known CAs such as Verisign, government bodies, or hardware manufacturers. Small variations can arise due to certificates added by the users through the system settings or by apps after requesting user permission. Mobile manufacturers such as HTC and Samsung[3] have alike additional certificates on their root store (e.g., *AddTrust*, *Deutsche Telekom*, *Sonera* and *U.S. Department of Defense*[4]) independently of the mobile operator, as shown in Figure 2. In this case, it is possible that hardware vendors add these certificates to the firmware images they build for their handsets. On the other hand, we find some roots, such as *CertiSign* and *ptt-post.nl*[5] exclusively on 60 to 70% of Motorola 4.1 devices, all of them subscribed to Verizon Wireless. Given that mobile operators have the ability to modify the firmware they ship on their subsidized handsets, Verizon Wireless may be the agent responsible for these certificates' presence. Similarly, we identified potential AT&T-specific inclusions on Motorola handsets, such as a *Microsoft Secure Server* certificate.

Certain handset-specific certificates are not used for TLS operations, such as the *GeoTrust CA for UTI* certificate (installed on Samsung 4.2 and 4.3 devices) or Motorola's certificates for FOTA (Firmware over-the-air) and SUPL (Secure User Plane Location). UTI (Unified Testing Initiative) is a non-profit group led by hardware vendors and mobile operators to improve the quality of mobile apps [2]. GeoTrust restricts the use of the certificate to members of the Java Verified Program [8]. The FOTA and SUPL certificates secure firmware updates and location-sensor assistance [28], respectively. Given the privacy-sensitive information transmitted over SUPL (including neighboring WiFi APs and cellular base stations)

[3] Note that Samsung 4.1 and 4.2 devices have similar root stores, but different to 4.3 and 4.4 ones, which are extended.

[4] The subject and issuer for this certificate is: *CN=DoD CLASS 3 Root CA,OU=PKI,OU=DoD,O=U.S. Government,C=US*. iOS7 also contains this certificate by default. It is not a WebTrust audited CA, so Mozilla considers it as an Intranet CA [25].

[5] Issued by the Dutch Postal Services, and also present in the Windows root store.

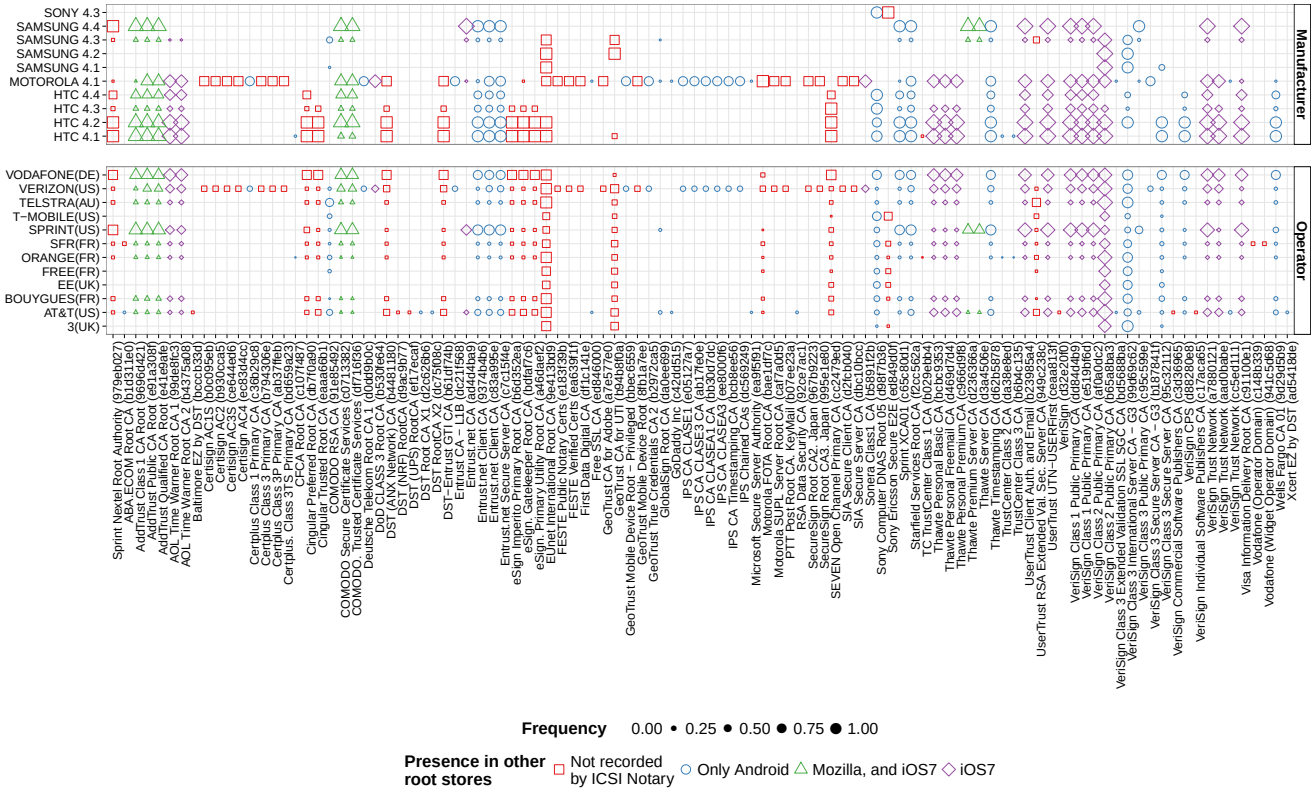

Figure 2: Scatterplot showing the ratio of sessions (size) with a given root certificate (x-axis) relative to the total number of sessions for a given operator or manufacturer (y-axis). The value in brackets shows the first 32 bits of the certificate subject.

and the security implications of software updates, these operations require a secure channel. Google, as well as mobile operators, Assisted GPS (A-GPS) chipset and handset vendors can also provide SUPL and FOTA services. Consequently, it is plausible that some of the root certificates included by other vendors may be used for similar purposes.

5.2 Additional observations

We observed several unusual root certificates for which we could not definitively establish their origin. These certificates, issued by 15 CAs, can be found in a wide range of handsets, mobile operators and countries. Here we group them into three categories and speculate regarding their possible origin:

- Certificates added manually by users for general connectivity management (VPN). We identified a number of self-signed certificates, each recorded exclusively on a single device.

- Certificates added by mobile operators for custom services and APIs such as location services, widgets, email or messaging. For these we observer *Vodafone Group* certificates, a *Verizon Wireless* certificate found on three Verizon Pantech[6] 4.1 devices, likely associated with the Verizon Network API [14]; *Meditel* certificates (a Moroccan ISP) recorded in several Samsung 4.1 devices in Bermuda; and two root certificates issued by *Telefonica* and recorded in Motorola 4.1 devices. These devices were connected to Telefonica/Movistar and Claro networks in Argentina, Colombia, Mexico and Peru. Presumably, the appearance of a root certificate issued by an operator different than the operator providing the network access suggests a user roaming or traveling abroad.

- Certificates issued by government agencies. These certificates are presumably legitimately installed by applications using Android APIs or directly by handset manufacturers. Two examples are the certificates issued by *Venezuelan National CA*—recorded in Compal devices in Taiwan—and the four certificates issued by the Chinese Finance CA (CFCA) found in HTC, Motorola and Lenovo devices from a number of countries such as Albania, Argentina, Belarus, Brazil, Bulgaria, China, Colombia, India, Mexico and Russia. The addition of CFCA certificates to Mozilla's root store program has spurred discussion [3].

These certificates did not manifest in the ICSI Notary data.

5.3 Root store validation with ICSI's Notary

Table 3 shows the total number of TLS certificates known to the Notary validated by each AOSP version and Mozilla root store. The results indicate that there few practical differences between them. The small variations mostly arise due to the fact some government-issued CA certificates (e.g., the U.S. DoD) do not appear in all root stores. However, not all certificates contribute equally to these numbers. Figure 3 shows the ECDF of the number of certificates (out of the more than 1 million non-expired certificates recorded by the Notary) that each root certificate can validate, grouped by different categories. We include the iOS7 and Mozilla root stores for comparison. The figure shows that the subset of AOSP certificates that are also included on Mozilla root store can validate most TLS

[6]Pantech manufactures phones for AT&T and Verizon in the USA.

Root store	No. validated certificates
Mozilla	744,069
iOS 7	745,736
AOSP 4.1	744,350
AOSP 4.2	744,350
AOSP 4.3	744,384
AOSP 4.4	744,398

Table 3: Number of certificates validated by Mozilla and AOSP root stores.

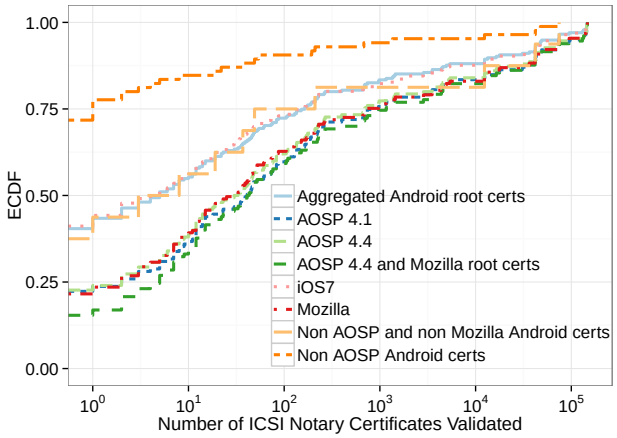

Figure 3: ECDF of the number of TLS certificates recorded by ICSI's Notary that each root store can progressively validate as we cumulatively consider each of its certificates (starting with the certificates that can validate the most additional certs).

sessions, while the superset of all Android root certificates collected by Netalyzr has a similar behavior as the iOS7 root store (the largest one).

An interesting observation concerns the y-axis offset in Figure 3 for each category. The offset indicates the percentage of certificates for a given group that did not validate any of the certificates recorded by the Notary for common operations such as HTTPS connections or TLS-secured email delivery. In particular, 23% of AOSP 4.4 root certificates did not validate any of the Notary TLS certificates, as opposed to 72% of the extra Android certificates that appear in neither the official AOSP 4.4 root store nor in Mozilla's. One could seemingly disable these certificates with little negative effect on the user experience or TLS functionality, though the Notary's does not provide a comprehensive enough perspective for us to state this definitively.

6. ROOTING ANDROID HANDSETS

In Section 2, we explained how rooting mobile handsets allows malicious mobile apps to add, delete and modify certificates on the root store. 24% of our Netalyzr sessions ran on rooted handsets. We identified certificates that appear exclusively in rooted handsets in 6% of those (so about 1.5% of all sessions).

Table 5 reports the set of root certificates more frequently appearing on rooted rather than non-rooted devices. None of these occurred in Notary traffic. Generally, the issuers of these certificates are the users themselves or small corporations looking to employ other means of secure communication such as VPNs. This is the case for the certificates signed by USER_X[7] and Mind Overflow

[7]Anonymized.

Root store category	Total root certs	Root certs that do not validate Notary certs
Non AOSP and Non Mozilla root certs	85	72%
Non AOSP root certs found on Mozilla's	16	38%
AOSP 4.4 and Mozilla root certs	130	15%
AOSP 4.1 certs	139	22%
AOSP 4.4 certs	150	23%
Aggregated Android root certs	235	40%
Mozilla root store certs	153	22%
iOS 7 root store certs	227	41%

Table 4: Number of root certificates found in ICSI's Notary per category, and how many of them did not validate any of the certificates stored on ICSI's Notary.

Certificate authority	Total devices
CRAZY HOUSE	70
MIND OVERFLOW	1
USER_X	1
CDA/EMAILADDRESS	1
CIRRUS, PRIVATE	1

Table 5: List of CAs and user self-signed certificates found more frequently on rooted devices.

(we could not identify any application or real CA with this name), both collected from the same device. Likewise, *CDA (Chaine de Distribution Alimentaire) Senegal* certificate was identified on a rooted Nexus 7 device connected from a Senegalese WiFi AP.

The certificate issued by Madkit-Crazy House (Ukraine) recorded in 70 different handsets illustrates the sort of security hazard that can result on rooted Android handsets. The ICSI Certificate Notary has no record of this certificate. After checking the applications running in the background on the affected handsets, we verified that this certificate is installed by the Freedom app [7], which enables free in-app purchases on the Google Play Store. This app requires root permissions (so it can modify the root store) and compels the user to accept egregious permissions such as accessing the Google accounts set up on the device, reading phone status and identity, and modifying system settings.

7. TLS INTERCEPTION

Netalyzr for Android checks the full trust chain of TLS connections to the domains of popular websites and mobile apps. We use the ICSI Notary to assess each of the observed certificates. Out of the 15K sessions, we identified a case of TLS interception for one user running a Nexus 7 device on Android 4.4, communicating with an HTTPS-proxied WiFi access point. In this case, a British marketing research provider called Reality Mine [11] signed the root certificates.

Looking deeper at the connectivity results reported by Netalyzr, we observe that this user's traffic flows via a `tun` interface that tunnels all traffic to a proxy controlled by Reality Mine (`v-us-49.analyzeme.me.uk`). The proxy name matches the name a Reality Mine app on Google Play [1]. In fact, all Android applications published by Reality Mine in Google Play (Consumer-Input Mobile, USA TouchPoints, MediaTrack, and AnalyzeMe) require (as of Oct. 2014) the user to accept both a permission for

Intercepted domains	Whitelisted domains
gmail.com:443	google-analytics.com:443
mail.google.com:443	maps.google.com:443
mail.yahoo.com:443	orcart.facebook.com:8883
orcart.facebook.com:443	play.google.com:443
www.bankofamerica.com:443	supl.google.com:7275
www.chase.com:443	www.facebook.com:443
www.hsbc.com:443	www.google.com:443
www.icsi.berkeley.edu:443	www.google.co.uk:443
www.outlook.com:443	www.twitter.com:443
www.skype.com:443	
www.viber.com:443	
www.yahoo.com:443	

Table 6: Domains being intercepted and whitelisted by Reality Mine HTTPS proxy.

changing handset network configuration and one to intercept user traffic. The latter permission, requested by any app behaving as a VPN client, allows the application to create a virtual `tun` interface with which it associates a raw read socket. In this case, we cannot identify whether TLS interception takes place on the local host versus the remote server, since the app could employ either approach.

In addition to network and traffic interception permissions, the application requires access to protected storage and the ability to read contacts, calendar, location, text messages, device ID, call information, Web bookmarks and history, and sensitive log data. The application requires an invitation from Reality Mine to run and does not require including any certificate in the root store.

The proxy listens on ports 80 and 443, intercepting and regenerating both root and intermediate certificates on-the-fly for specific domains. The proxy whitelists Google's SUPL service (7275) and Facebook chat (8883), as well as popular apps implementing mechanisms like certificate pinning for security [19, 21]: Facebook, Twitter, and most Google services. Table 6 summarizes this observation. We contacted Reality Mine to verify whether their TLS interception occurs with user consent. According to their statements, they recruit participants and all data is cleaned and anonymized in line with the best practices defined by Esomar association [6]. As of this writing, we have not received a response we requested from Reality Mine regarding the terms of use and the privacy considerations of app users.

8. RECOMMENDATIONS

Given that Android does not differentiate the use and scope of certificates installed in the root store, and does not protect the root store from malicious apps with root permissions, a bloated root store increases the attack surface for malicious apps. Such design decisions make mobile users more vulnerable to TLS interception attacks. In order to improve mobile user security, we recommend enforcing an audited and more strict root store for Android, per the approaches adopted by Mozilla and iOS. If operators wish to install additional certificates, they should do so via this auditing process.

We question whether users have sufficient awareness of the consequences of their actions on their own security and privacy. Our results highlight how rooting an Android device can seriously undermine user protections. As widely reported in previous work, users must exercise prudence and caution when granting rights to applications during initial installation and subsequent updates. App developers can mask malicious intentions behind seemingly helpful permission requests such as traffic interception to enable VPNs, which, if granted, provide access to a user's entire traffic. In effect, abusing such app permissions breaks the app sandboxing model implemented in Android to protect users' privacy.

9. SUMMARY

It is generally recognized that certificate root stores represent a significant potential weak link in TLS security. X.509 certificates found in the root store are automatically considered secure and trustworthy: users will trust any CA that their mobile device and its apps trusts. In this work we used data from Netalyzr for Android and the ICSI Certificate Notary to characterize the root stores of thousands of Android phones in the wild. Our analysis highlights three concerns. First, we demonstrate how mobile operators and hardware manufacturers modify and extend Google's official Android root store. Second, we report on how rooted Android handsets can lead to undermining the model of audited and supervised root stores, allowing malicious apps to modify the store without user awareness. Third, we illustrate the use of TLS interception for marketing research. We show how apps can request overreaching permissions to exploit the lack of security mechanisms such as certificate pinning. These observations lead us to highlight the need for an audited and strictly controlled root store for Android, and efforts towards improving user awareness of the implications of their actions when granting permissions.

10. ACKNOWLEDGMENTS

We are deeply grateful to *Netalyzr*'s many users for making this study possible, and also for their helpful feedback. We would like to thank the anonymous reviewers for their valuable comments. This work was supported by the National Science Foundation under grants NSF-0831535, NSF-1213157, and NSF-1237265, and by the DHS Directorate of Science and Technology under grant N66001-12-C-0128. We also wish to thank Amazon, Comcast, and Google for their generous support.

11. REFERENCES

[1] AnalyzeMe App. https://play.google.com/store/apps/details?id=com.apadmi.analyzeme.android.

[2] App Quality Alliance. http://www.appqualityalliance.org.

[3] Bugzilla @ Mozilla. CFCA root CA. https://bugzilla.mozilla.org/show_bug.cgi?id=926029.

[4] Convergence. http://convergence.io.

[5] EFF Observatory. https://www.eff.org/observatory.

[6] Esomar. http://www.esomar.org.

[7] Freedom App. http://system.in-appstore.com/freedom.

[8] Geotrust Unified Testing Initiative (UTI) Root Key Certificate. https://www.geotrust.com/resources/root-certificates/root12_terms.html.

[9] Netalyzr. http://netalyzr.icsi.berkeley.edu.

[10] Netalyzr for Android. Google Play. https://play.google.com/store/apps/details?id=edu.berkeley.icsi.netalyzr.android.

[11] Reality Mine. http://www.realitymine.com.

[12] Security with HTTPS and SSL. http://developer.android.com/training/articles/security-ssl.html.

[13] The ICSI Certificate Notary. http://notary.icsi.berkeley.edu.

[14] Verizon Network API. http://developer.verizon.com/content/vdc/en/verizon-tools-apis/verizon_apis/network-api.html.

[15] X509TrustManager Android API. http://developer.android.com/reference/javax/net/ssl/X509TrustManager.html.

[16] B. Amann, R. Sommer, M. Vallentin, and S. Hall. No Attack Necessary: The Surprising Dynamics of SSL Trust Relationships. In *Proc. of ACM ACSAC*, 2013.

[17] B. Amann, M. Vallentin, S. Hall, and R. Sommer. Extracting Certificates from Live Traffic: A Near Real-Time SSL Notary Service. Technical Report TR-12-014, ICSI, November 2012.

[18] C. Amrutkar, P. Traynor, and P. C. van Oorschot. Measuring SSL Indicators on Mobile Browsers: Extended Life, or End of the Road? In *Proc. Inf. Sec. Conf.*, 2012.

[19] Facebook. Secure browsing by default. https://www.facebook.com/notes/facebook-engineering/secure-browsing-by-default/10151590414803920.

[20] S. Fahl, M. Harbach, T. Muders, L. Baumgärtner, B. Freisleben, and M. Smith. Why Eve and Mallory Love Android: An Analysis of Android SSL (in)Security. In *ACM CCS*, 2012.

[21] M. Georgiev, S. Iyengar, S. Jana, R. Anubhai, D. Boneh, and V. Shmatikov. The Most Dangerous Code in the World: Validating SSL Certificates in Non-browser Software. In *Proc. ACM CCS*, 2012.

[22] T. Ho, D. Dean, X. Gu, and W. Enck. Prec: practical root exploit containment for android devices. In *Proc. ACM CODASPY*, 2014.

[23] R. Housley, W. Ford, W Polk, and D. Solo. Rfc 5280: Internet X. 509 Public Key Infrastructure Certificate and CRL profile, 2008.

[24] L. Huang, A. Rice, E. Ellingsen, and C. Jackson. Analyzing Forged SSL Certificates in the Wild. In *Proc. IEEE Symposium on Security and Privacy*, 2014.

[25] Mozilla Foundation. Bug 208323 - Add DoD root CA. https://bugzilla.mozilla.org/show_bug.cgi?id=208323.

[26] H. Perl, S. Fahl, and M. Smith. You Won't Be Needing These Any More: On Removing Unused Certificates From Trust Stores. In *Financial Cryptography and Data Security*, 2014.

[27] B. Schneier. New NSA Leak Shows MITM Attacks Against Major Internet Services. https://www.schneier.com/blog/archives/2013/09/new_nsa_leak_sh.html, 2014.

[28] N. Vallina-Rodriguez, J. Crowcroft, A. Finamore, Y. Grunenberger, and K. Papagiannaki. When Assistance Becomes Dependence: Characterizing the Costs and Inefficiencies of A-GPS. *ACM SIGMOBILE Mob. Comput. Commun. Rev.*, 2013.

[29] D. Wendlandt, D. Andersen, and A. Perrig. Perspectives: Improving SSH-style Host Authentication with Multi-Path Probing. In *USENIX Annual Technical Conference*, 2008.