Studying TLS Usage in Android Apps

Abbas Razaghpanah, Arian Akhavan Niaki, Narseo Vallina-Rodriguez,
Srikanth Sundaresan, Johanna Amann, Phillipa Gill

Paper: ACM CoNext 2017

Encryption is Everywhere

However…

- TLS is also an important component of mobile applications
  - 88% of Android applications use TLS
- Unlike Web browsers and servers…
  - …many application developers implementing TLS
  - …many opportunities to make errors!

Understanding TLS on Android

- Understanding of TLS on Android has been limited …
- Static analysis: Explores all code paths, but not neccessarily those taken in practice
- Dynamic analysis: May not cover all code paths

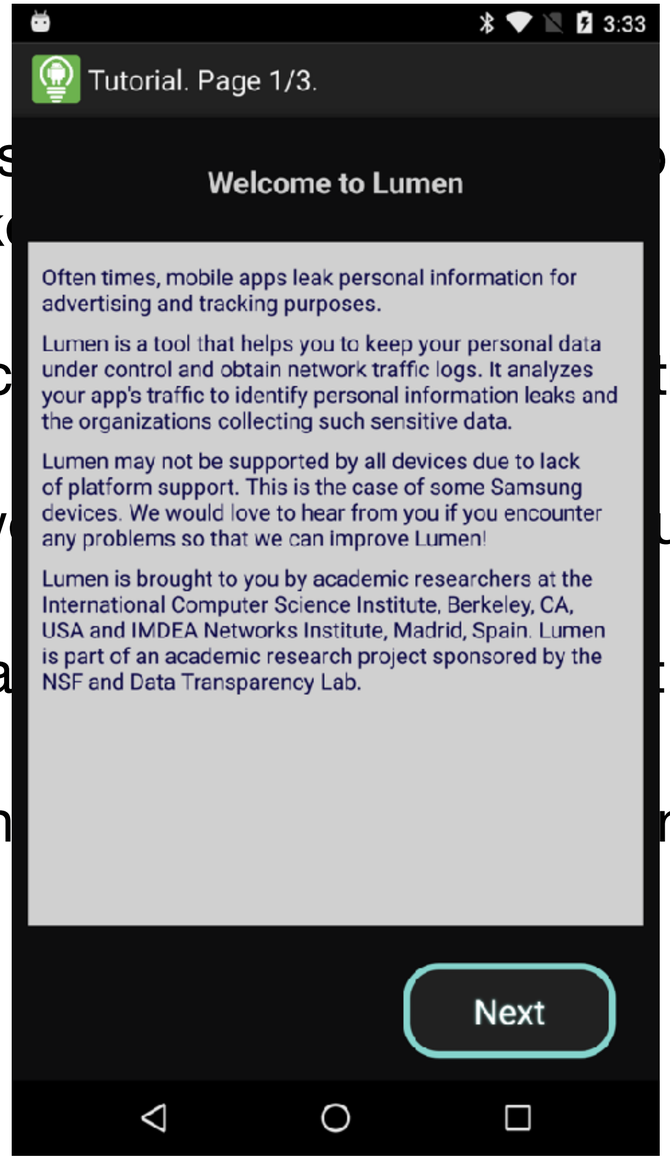
- Our Solution: **Lumen**
- User space traffic monitoring on Android
- Crowd source measurements of application behavior
- Collect anonymized TLS handshake data between apps and servers

…Wait a minute

- Our study is deemed to be non-human-subject research by UC Berkeley's IRB
- We collect no private information of traffic (encrypted or unencrypted)
- All web browser traffic is excluded
- We are studying software, not people
- We have a comprehensive consent process in place

…Wait a minute

- Our s…n-human-subject research by UC Berk…
- We c…tion of traffic (encrypted or unencrypted)
- All w…uded
- We a… people
- We h…nsent process in place

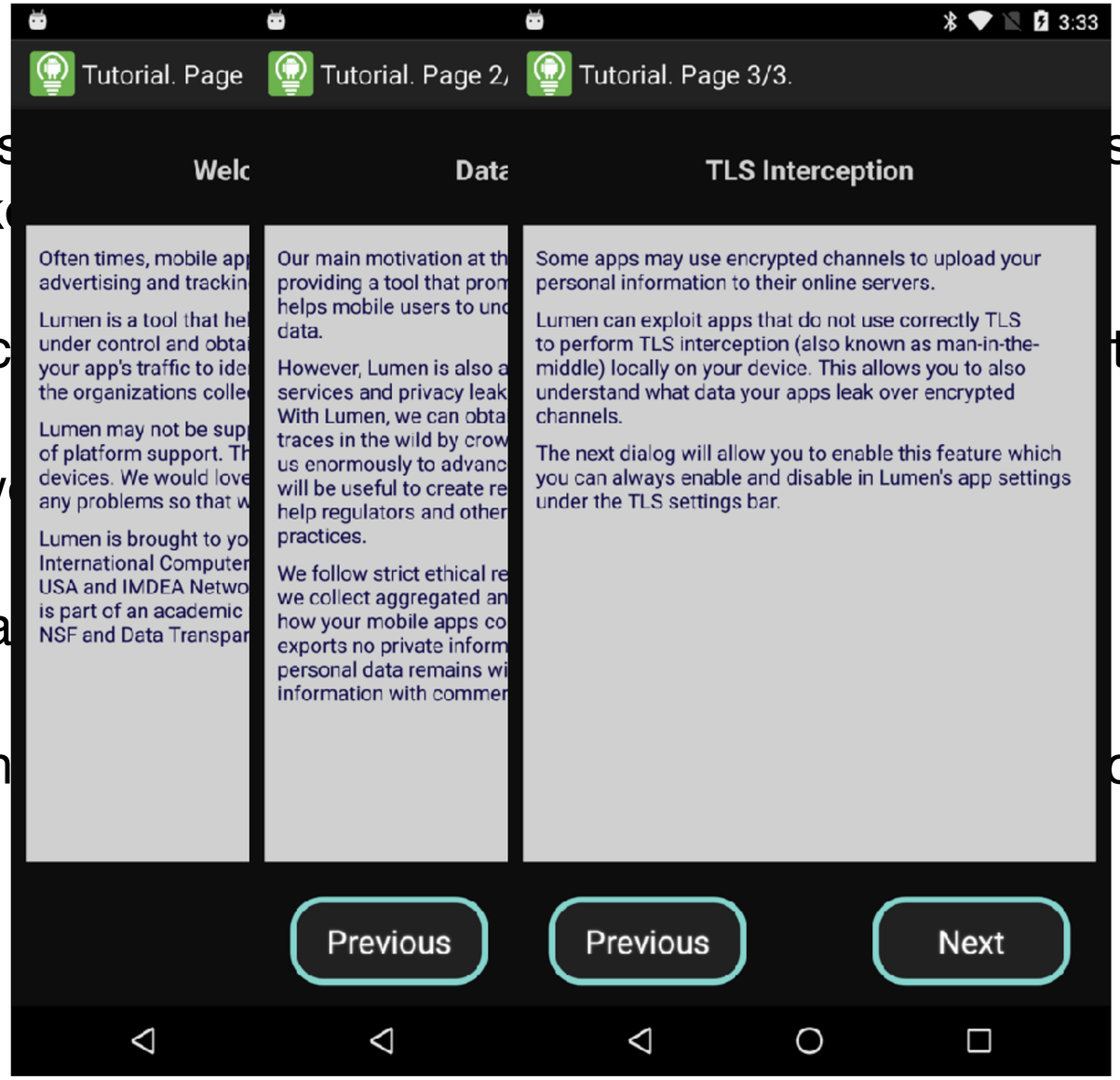


…Wait a minute

- Our s -subject research by UC Berk
- We c ffic (encrypted or unencrypted)
- All w
- We a
- We h cess in place

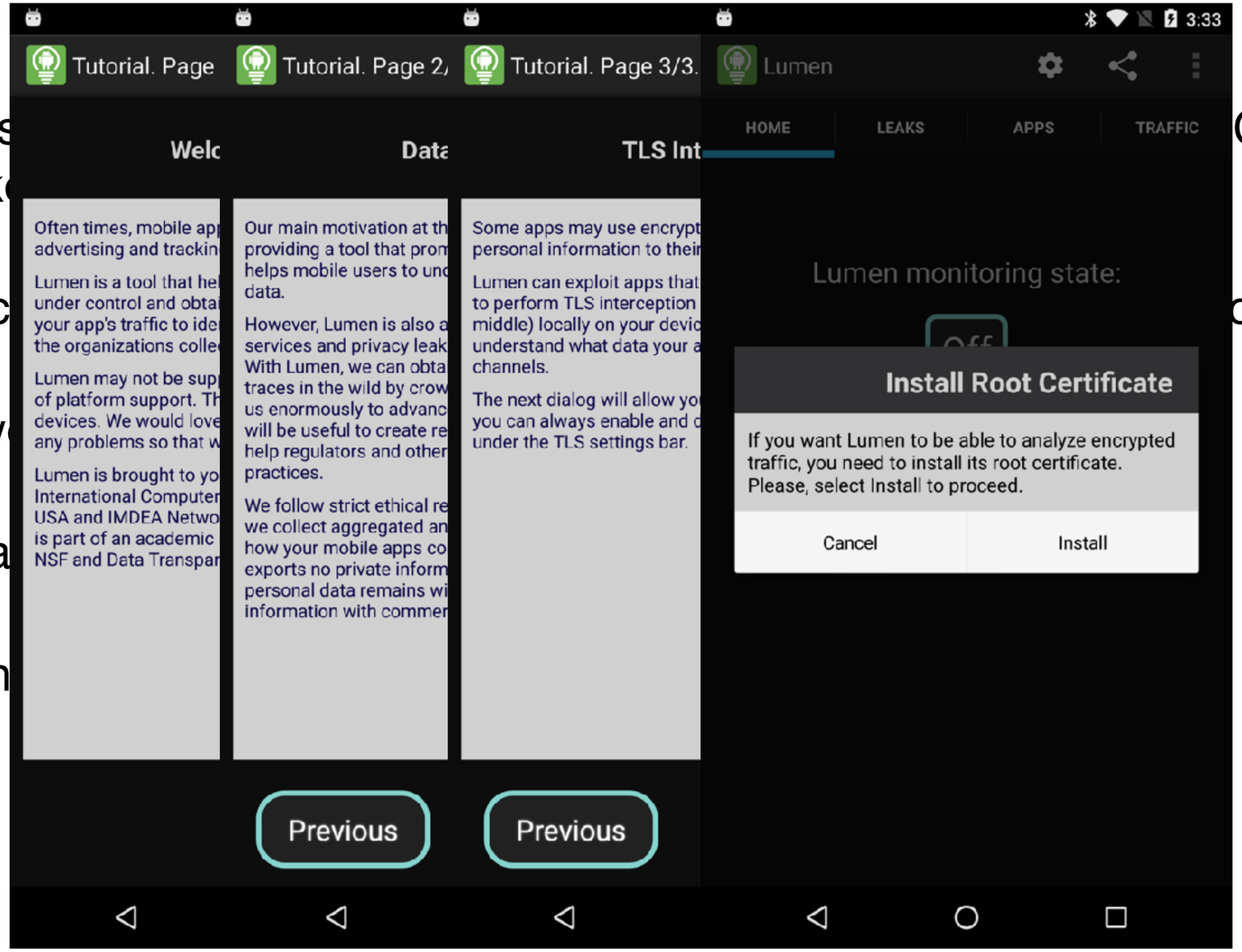
…Wait a minute

- Our s search by UC Berk
- We c ted or unencrypted)
- All w
- We a
- We h ce



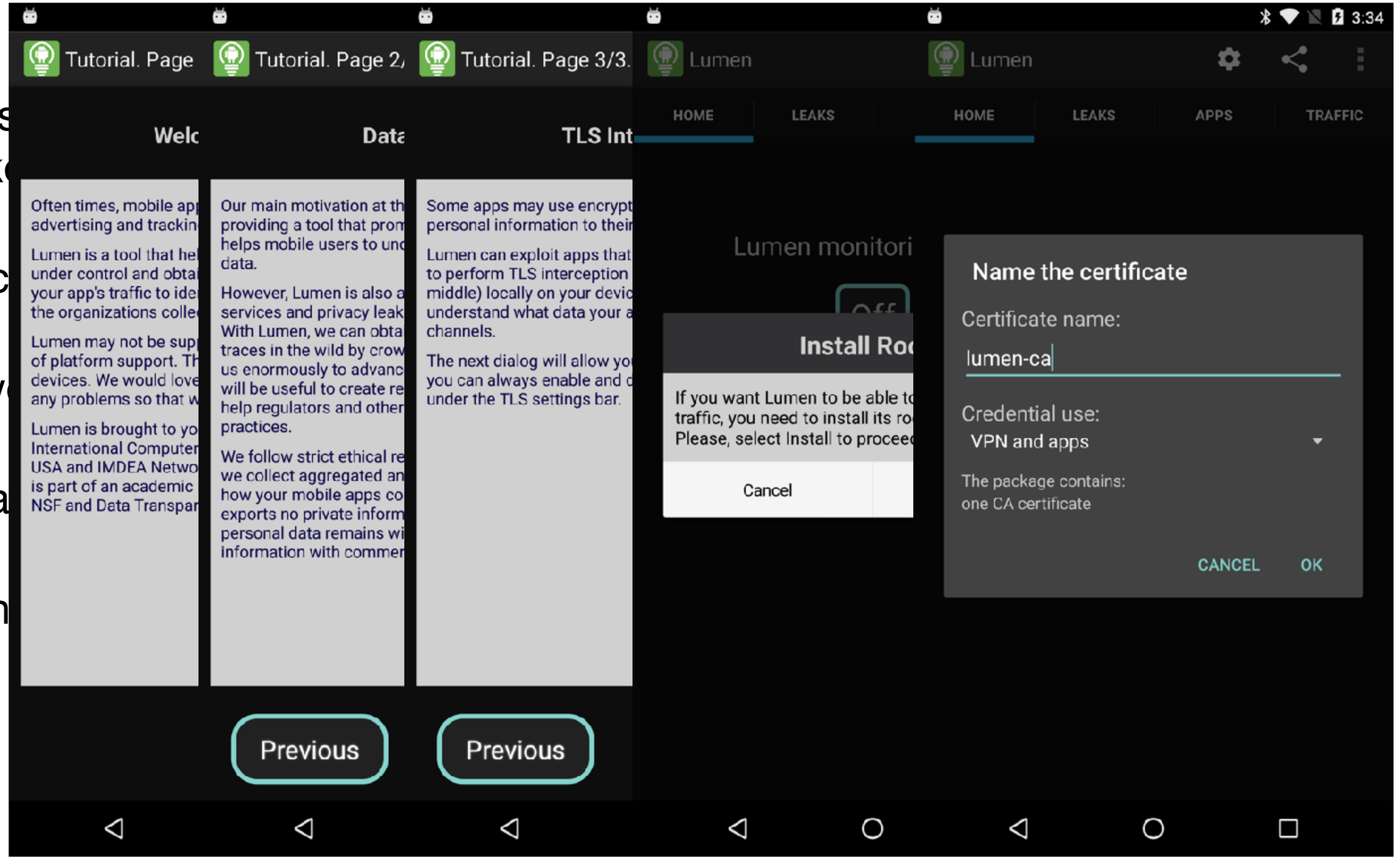
…Wait a minute

- Our s… C Berk…
- We c… crypted)
- All w…
- We a…
- We h…



…Wait a minute

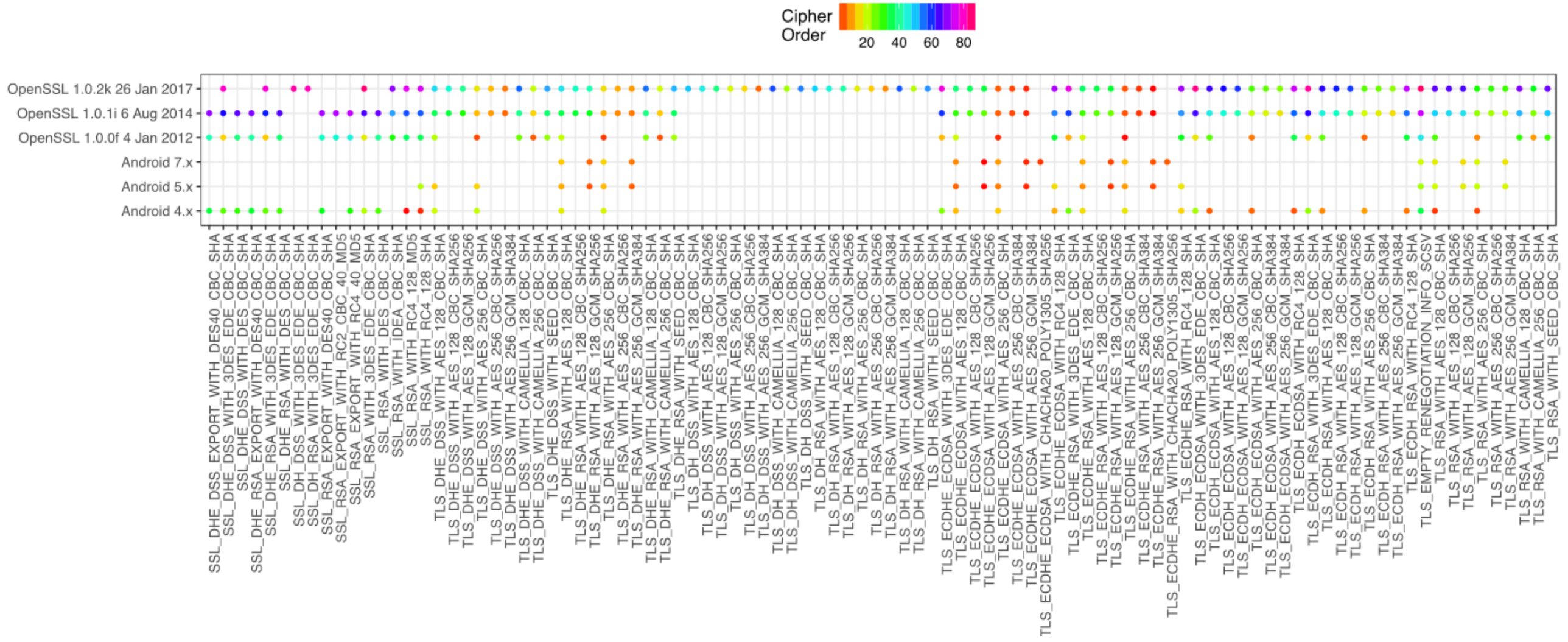
- Our s
  Berk
- We c
- All w
- We a
- We h



What do we collect?

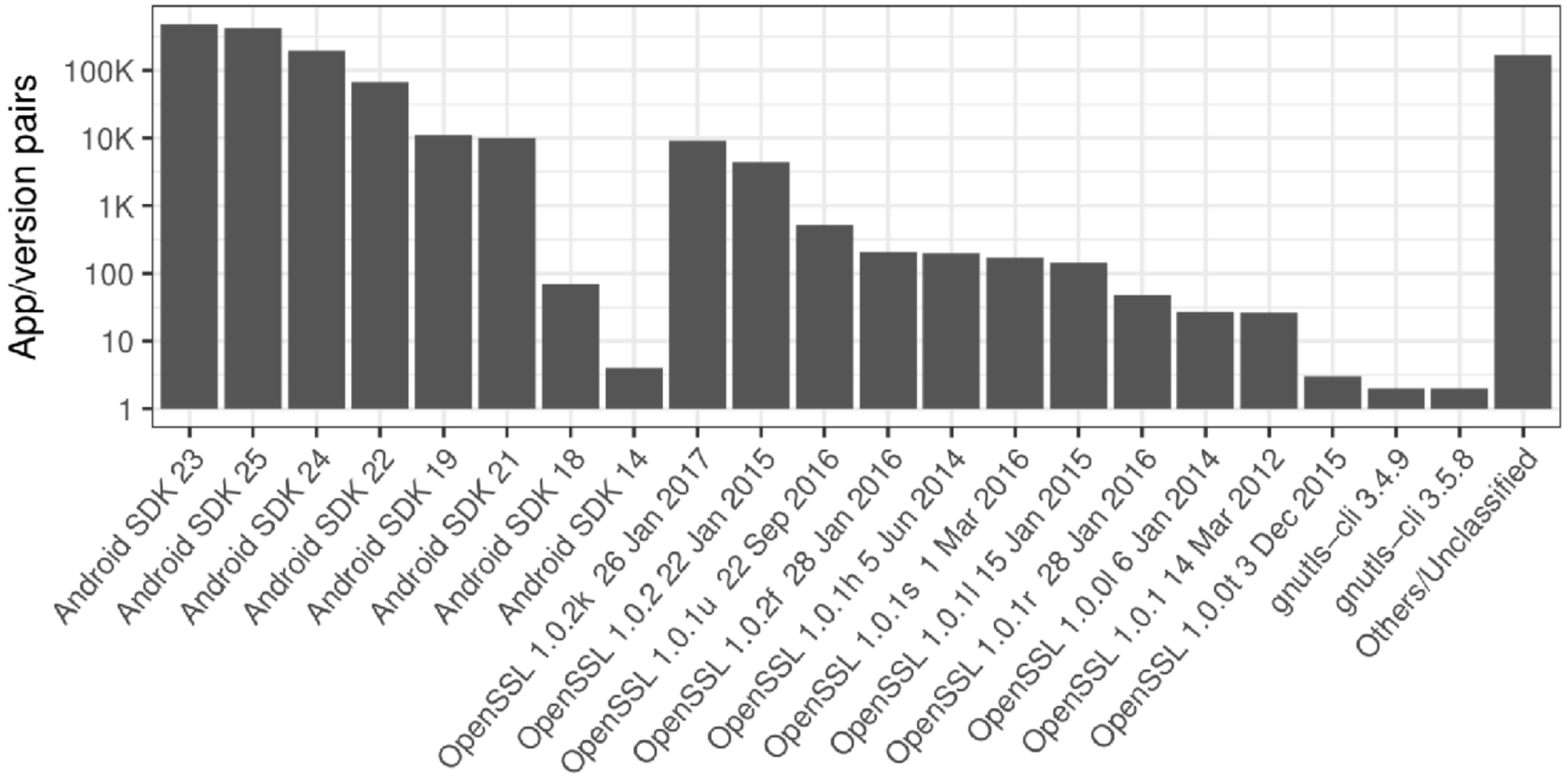
- Three key items:
  - Client Hello
  - Server Hello (with certificates)
  - Failures of our TLS proxy (reveals pinning)

Users	>5,000 from >100 countries
Connections (11/15—6/17)	1,486,082
Apps	7,258
Domains (unique SNIs)	34,176
TCP ports	250
Unique device/OS combos	891
TLS proxy failures	684,209 (4,268 apps and 10,753 domains)


Cipher Order
20
40
60
80
OpenSSL 1.0.2k 26 Jan 2017
OpenSSL 1.0.1i 6 Aug 2014
OpenSSL 1.0.0f 4 Jan 2012
Android 7.x
Android 5.x
Android 4.x
SSL_DHE_DSS_EXPORT_WITH_DES40_CBC_SHA
SSL_DHE_DSS_WITH_3DES_EDE_CBC_SHA
SSL_DHE_DSS_WITH_DES_CBC_SHA
SSL_DHE_RSA_EXPORT_WITH_DES40_CBC_SHA
SSL_DHE_RSA_WITH_3DES_EDE_CBC_SHA
SSL_DHE_RSA_WITH_DES_CBC_SHA
SSL_DH_DSS_WITH_3DES_EDE_CBC_SHA
SSL_DH_DSS_WITH_DES_CBC_SHA
SSL_DH_RSA_WITH_3DES_EDE_CBC_SHA
SSL_DH_RSA_WITH_DES_CBC_SHA
SSL_RSA_EXPORT_WITH_DES40_CBC_SHA
SSL_RSA_EXPORT_WITH_RC2_CBC_40_MD5
SSL_RSA_EXPORT_WITH_RC4_40_MD5
SSL_RSA_WITH_3DES_EDE_CBC_SHA
SSL_RSA_WITH_DES_CBC_SHA
SSL_RSA_WITH_IDEA_CBC_SHA
SSL_RSA_WITH_RC4_128_MD5
SSL_RSA_WITH_RC4_128_SHA
TLS_DHE_DSS_WITH_AES_128_CBC_SHA
TLS_DHE_DSS_WITH_AES_128_CBC_SHA256
TLS_DHE_DSS_WITH_AES_128_GCM_SHA256
TLS_DHE_DSS_WITH_AES_256_CBC_SHA
TLS_DHE_DSS_WITH_AES_256_CBC_SHA256
TLS_DHE_DSS_WITH_AES_256_GCM_SHA384
TLS_DHE_DSS_WITH_CAMELLIA_128_CBC_SHA
TLS_DHE_DSS_WITH_CAMELLIA_256_CBC_SHA
TLS_DHE_DSS_WITH_SEED_CBC_SHA
TLS_DHE_RSA_WITH_AES_128_CBC_SHA
TLS_DHE_RSA_WITH_AES_128_CBC_SHA256
TLS_DHE_RSA_WITH_AES_128_GCM_SHA256
TLS_DHE_RSA_WITH_AES_256_CBC_SHA
TLS_DHE_RSA_WITH_AES_256_CBC_SHA256
TLS_DHE_RSA_WITH_AES_256_GCM_SHA384
TLS_DHE_RSA_WITH_CAMELLIA_128_CBC_SHA
TLS_DHE_RSA_WITH_CAMELLIA_256_CBC_SHA
TLS_DHE_RSA_WITH_SEED_CBC_SHA
TLS_DH_DSS_WITH_AES_128_CBC_SHA
TLS_DH_DSS_WITH_AES_128_CBC_SHA256
TLS_DH_DSS_WITH_AES_128_GCM_SHA256
TLS_DH_DSS_WITH_AES_256_CBC_SHA
TLS_DH_DSS_WITH_AES_256_CBC_SHA256
TLS_DH_DSS_WITH_AES_256_GCM_SHA384
TLS_DH_DSS_WITH_CAMELLIA_128_CBC_SHA
TLS_DH_DSS_WITH_CAMELLIA_256_CBC_SHA
TLS_DH_DSS_WITH_SEED_CBC_SHA
TLS_DH_RSA_WITH_AES_128_CBC_SHA
TLS_DH_RSA_WITH_AES_128_CBC_SHA256
TLS_DH_RSA_WITH_AES_128_GCM_SHA256
TLS_DH_RSA_WITH_AES_256_CBC_SHA
TLS_DH_RSA_WITH_AES_256_CBC_SHA256
TLS_DH_RSA_WITH_AES_256_GCM_SHA384
TLS_DH_RSA_WITH_CAMELLIA_128_CBC_SHA
TLS_DH_RSA_WITH_CAMELLIA_256_CBC_SHA
TLS_DH_RSA_WITH_SEED_CBC_SHA
TLS_ECDHE_ECDSA_WITH_3DES_EDE_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256
TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
TLS_ECDHE_RSA_WITH_RC4_128_SHA
TLS_ECDH_ECDSA_WITH_3DES_EDE_CBC_SHA
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA
TLS_ECDH_ECDSA_WITH_AES_128_CBC_SHA256
TLS_ECDH_ECDSA_WITH_AES_128_GCM_SHA256
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA
TLS_ECDH_ECDSA_WITH_AES_256_CBC_SHA384
TLS_ECDH_ECDSA_WITH_AES_256_GCM_SHA384
TLS_ECDH_ECDSA_WITH_RC4_128_SHA
TLS_ECDH_RSA_WITH_3DES_EDE_CBC_SHA
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA
TLS_ECDH_RSA_WITH_AES_128_CBC_SHA256
TLS_ECDH_RSA_WITH_AES_128_GCM_SHA256
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA
TLS_ECDH_RSA_WITH_AES_256_CBC_SHA384
TLS_ECDH_RSA_WITH_AES_256_GCM_SHA384
TLS_ECDH_RSA_WITH_RC4_128_SHA
TLS_EMPTY_RENEGOTIATION_INFO_SCSV
TLS_RSA_WITH_AES_128_CBC_SHA
TLS_RSA_WITH_AES_128_CBC_SHA256
TLS_RSA_WITH_AES_128_GCM_SHA256
TLS_RSA_WITH_AES_256_CBC_SHA
TLS_RSA_WITH_AES_256_CBC_SHA256
TLS_RSA_WITH_AES_256_GCM_SHA384
TLS_RSA_WITH_CAMELLIA_128_CBC_SHA
TLS_RSA_WITH_CAMELLIA_256_CBC_SHA
TLS_RSA_WITH_SEED_CBC_SHA

TLS Library Usage

- 84% of application versions in our dataset use OS-default libraries with default settings



Why do Apps not use defaults?

- To improve security:
  - Facebook uses OpenSSL and removes weaker cipher suites from the list (e.g. RC4 and 3DES ciphers); it also uses Facebook-specific ALPN
  - Twitter uses OS-provided libraries with a reordered cipher suite list

- Some do it wrong:
  - Some private messaging and VoIP apps use their own short cipher suite lists that do not have any forward-secret ciphers

- Others use third party libraries instead of the default:
  - Firefox uses NSS, VLC & SoundCloud GnuTLS (some versions)

Weak/outdated primitives

- SSLv3:
  - Supported by any app running on Android 5.0 and below (more than 61% of phones)
  - EA Games apps (FIFA Mobile, Madden NFL Mobile, etc.) with 100s of millions of installs, even when running on versions of Android that do not support it by default

- Null and Anonymous ciphers
  - Apps like TuneIn Radio with hundreds of millions of installs
  - Multiple EA games

- Export-grade ciphers:
  - Android 4.0 and below
  - Tiffany Alvord Dream World, a children’s game that has over one million installs

- Most apps with weak ciphers use poorly-configured OpenSSL

Solutions?

- De-couple TLS updates from OS updates!
  - TLS should be able to updated independent of the rest of the firmware, making it easier to update without manufacturer/vendor cooperation
  - Google is already doing this with Google Play Services (which bundle their own TLS library and certificate stores), so why not do the same with the OS-provided TLS library?
- Give more configuration options to developers
  - This way apps that need extra configuration options (e.g. setting ALPNs) are not forced to use something else

Certificates and Trust

- Android root stores often have “impurities” [Vallina-Rodriguez et al.]
- Some apps do not trust these trust stores and bundle their own CA certificates, pin server certificates, or use self-signed certificates
  - E.g. Firefox (bundles CA cert. store), Uber, Google, Paypal, Facebook (certificate pinning), Yandex (bundles unofficial Yandex root CA), Samsung apps (self-signed certs.) etc.
  - Implemented poorly, these can open up apps to MITM attacks

What does this all mean?

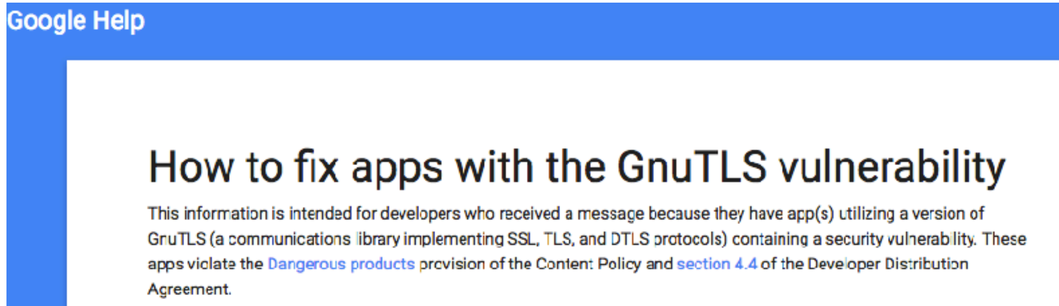
- Most apps (98%) trust OS-provided CA stores, and are vulnerable to MITM attacks when those are polluted

- Some apps pin certificates to mitigate the problem of polluted CA stores
  - It is not very prevalent (less than 2% of apps)
  - This can be problematic when done poorly: major system recovery app (with root access and ability to flash system firmware, bootloader, recovery, etc.) downloads CA bundle from the cloud in the clear

How do we fix it?

- What do we do with all the polluted CA certificate stores?
  - Google needs to ensure (e.g. through Android's licensing terms) that vendors can not surreptitiously inject their own CA certificates in trust stores
  - CA certificates also need to be able to be updated independently

- But some will still use their own libraries and pin certificates…
  - Make sure developers are properly educated about TLS
  - Detect and prevent poor implementations
  - Google has done something similar in the past: they implemented a tool that prevented developers from uploading apps that used a vulnerable version of GnuTLS and informed them about the issue



Summary

- First study of TLS usage in Android apps at scale
- Majority of apps (84%) use OS-provided libraries with default settings
- Apps using OS-defaults are vulnerable when the OS is outdated
- Apps using 3rd-party libraries and configurations are prone to misconfiguration and are therefore vulnerable
- Found low use of certificate pinning and CA bundling (less than 2%)
- Provided insights and potential solutions to the problems we found