SICUREZZA IN AMBITO MOBILE

Vincenzo Calabrò

Mobile Security?

Mobile Security = Mobile Device Security

- Sicurezza degli smartphone

Quindi:

- Quanto visto fino ad ora e in aggiunta:
  - Sicurezza delle App
  - Vulnerabilità specifiche di uno smartphone (tipo di connessione, tipo di utilizzo, tipologia di dati contenuti e trasmessi, ecc.)

Però possiamo fargli comprendere anche:

- Sicurezza Internet Of Things (IoT)
- Sicurezza del Cloud

Mobile Security vs PC Security?

Risorse limitate

- Batteria, CPU, memoria, banda
- Quindi: non sempre le soluzioni valide per i PC si possono applicare ai dispositive mobili

Più interfacce per la connessione

- Bluetooth, infrarossi, WiFi, reti cellulari, USB, tethering

Portatile/Mobile

- Soggetto alla perdita/furto
- Spesso attacchi short range (WiFi, Bluetooth)

Numero più elevato di software (app)

- Vendute/distribuite da terze parti non fidate

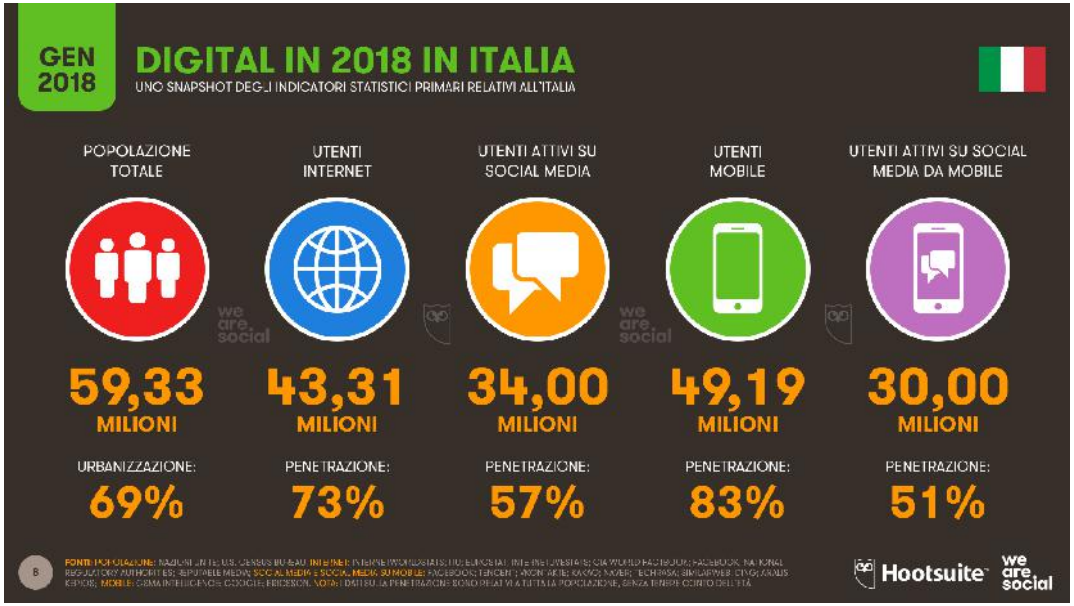
Perché Mobile Security vs PC Security?

User education

- Un numero più elevato di utenti e meno esperti

Ubiquitous

- Più economico, quindi più diffuso



Fonte: `https://wearesocial.com`

Contiene (molte!) informazioni sensibili

- Posizione
- Pseudo-identità (MAC, indirizzo Bluetooth, IMEI, IMSI)
- App per pagamenti, home-banking, carte di credito

Mobile threats? (1) – Obiettivi dell'attacco

Quali dati ci sono nel cellulare?

- Dati bancari e/o app finanziarie
- Credenziali per l'autenticazione
- Dati personali: foto, video, messaggi di posta, sms, contatti …
- Dati scambiati in (e relativi a) social network
- Musica, film, …
- Log delle attività (calendario, chiamate, ecc.)
- Informazione sulla posizione e storia degli spostamenti
- Accesso a dati su cloud o servizi

Mobile threats? (2) – Obiettivi dell'attacco

Specifici dei cellulari:

- Identificare la posizione
- Registrare chiamate telefoniche o flusso di chiamate
- Accedere ai messaggi
- Chiamate/SMS a numeri a tariffazione maggiorata di proprietà dell'attaccante ma registrati anonimamente
- Consumare tutta la batteria del cellulare

Simili ai PC:

- Malware
- Phishing
- Malvertising (malicious advertising)

Mobile threats? (3) – Tipologie di attacco?

Attaccante con accesso fisico al dispositivo

- Cerca di sbloccare il telefono
- Sfrutta vulnerabilità note per bypassare il blocco

Attacchi al sistema

- Sfrutta vulnerabilità della *piattaforma* o dei *protocolli di comunicazione* iniettando malware tramite download da web, dati malformati, ecc.

Attacchi alle App

- Utilizzare app maliziose o vulnerabili per rubare dati, utilizzare in modo sbagliato il sistema, attaccare altre app

Mobile threats? (4) – Vettori di attacco?

- Browser Web
- Sistema operativo
- Link maliziosi su siti web e/o social network
  - **Malware**
- Smartphone-based:
  - App (Google Market vs App Store)
    - **Malware**
  - SMS/MMS
  - Canale di comunicazione: WiFi, Blutooth, GSM

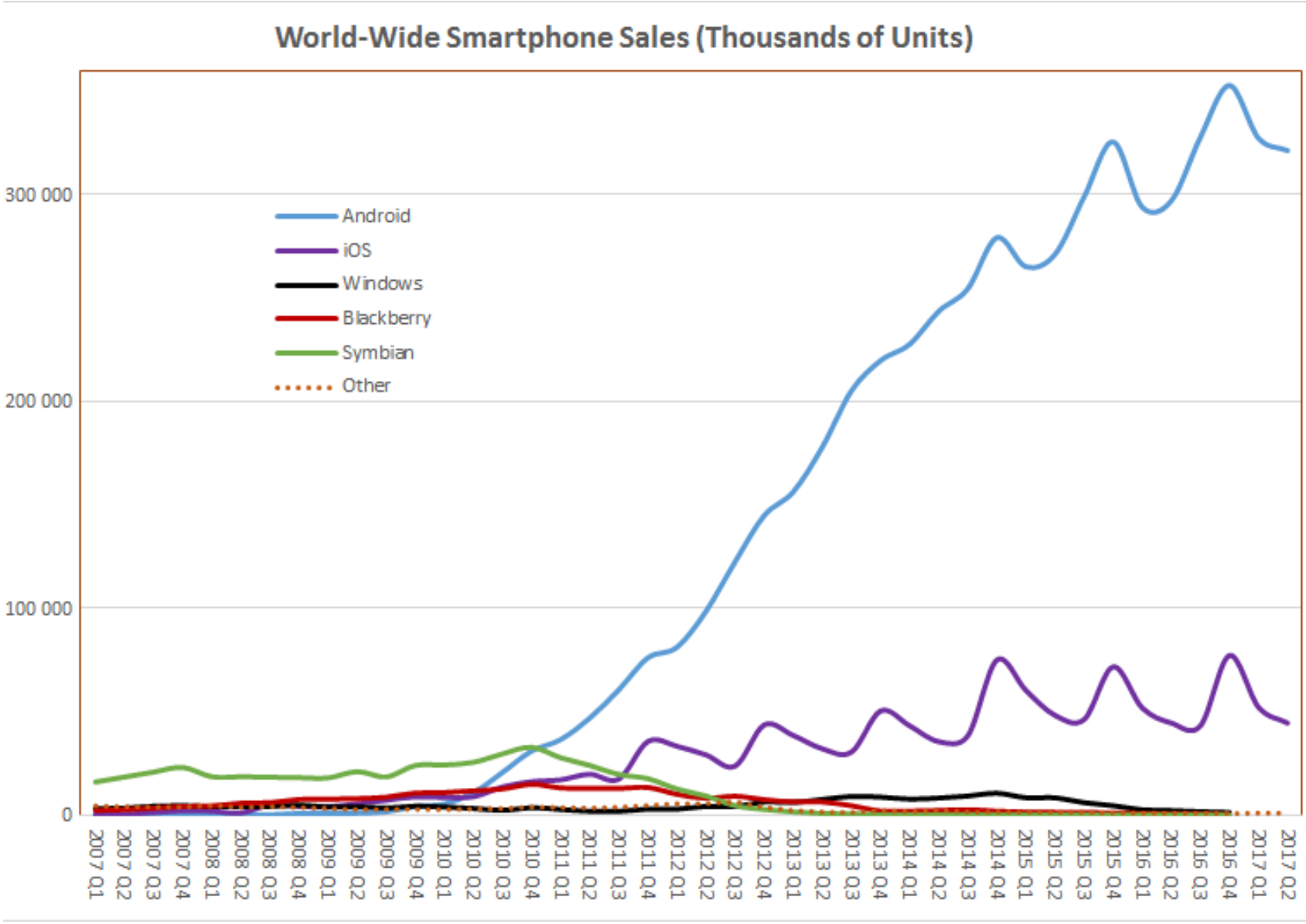
Mobile threats? (5) – Contromisure?

Aspetti di sicurezza che deve considerare una piattaforma/SO mobile:

- Accesso sicuro al dispositivo
- Sicurezza del sistema
- Sicurezza delle App
- Sicurezza dei dati (sul cellulare e su cloud)

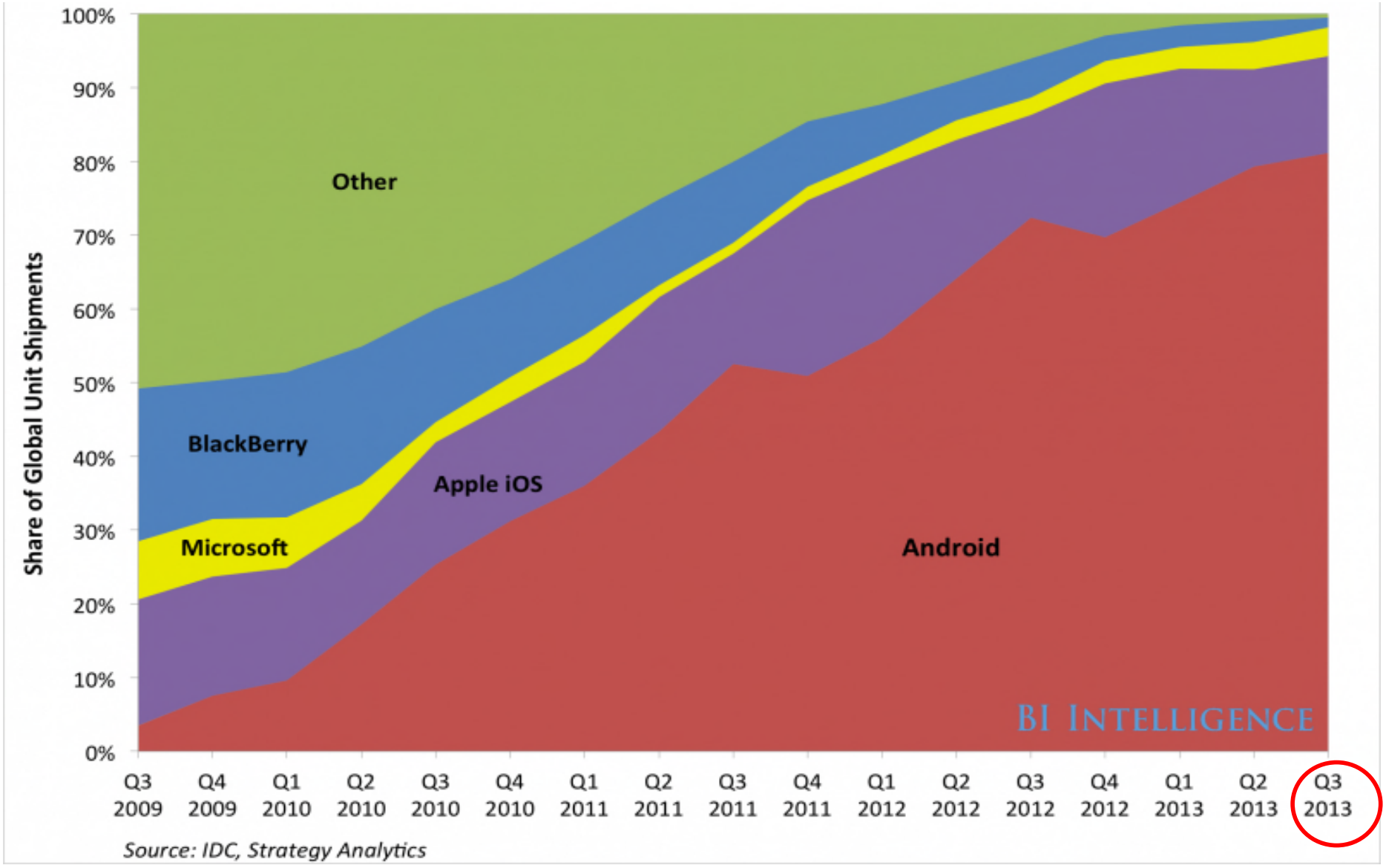
Quale smartphone attaccare? (1)

Cellulari (OS): numero di dispositivi acquistati



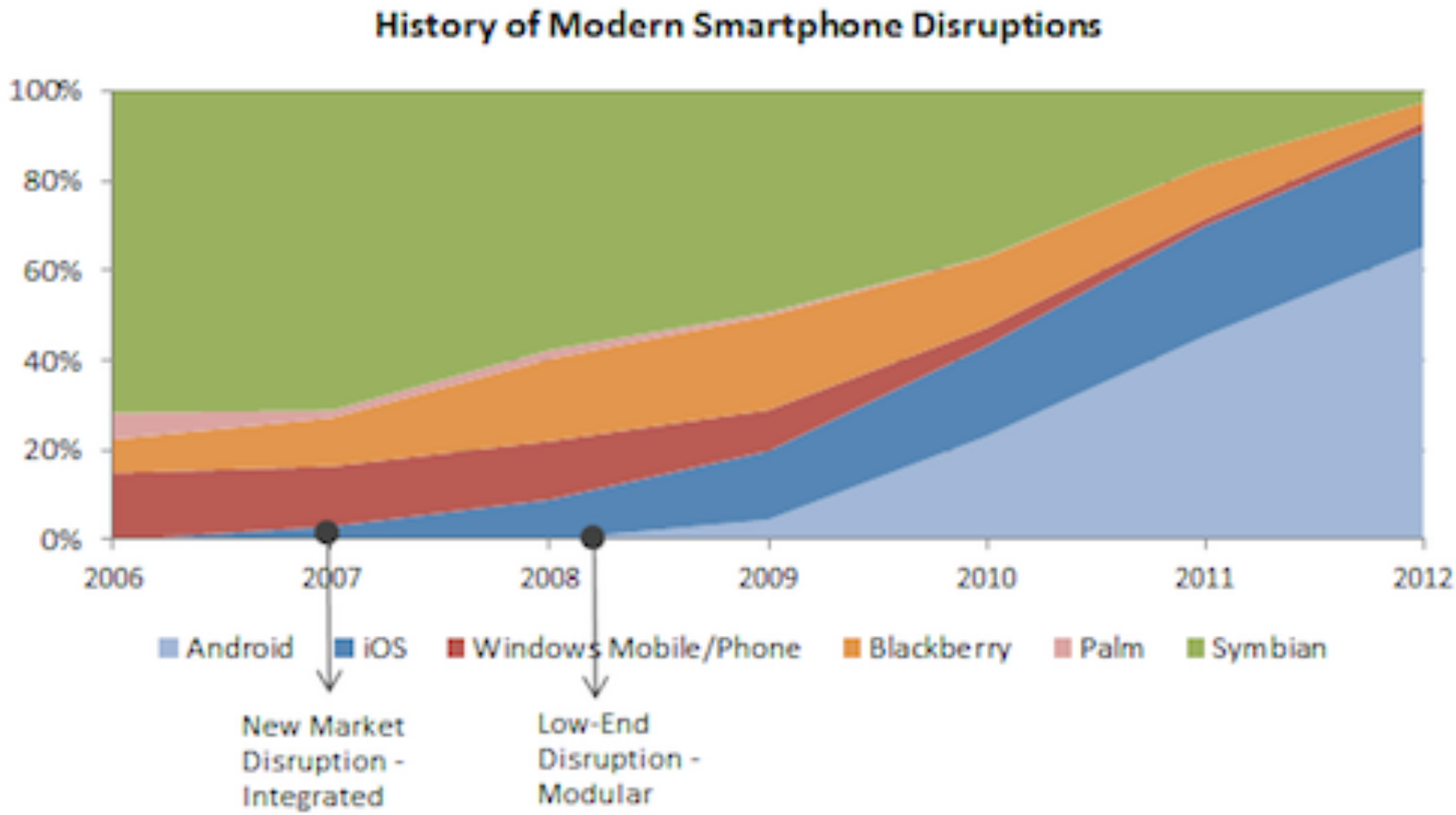
Quale smartphone attaccare? (2)

Cellulari (OS): quota di mercato mondiale



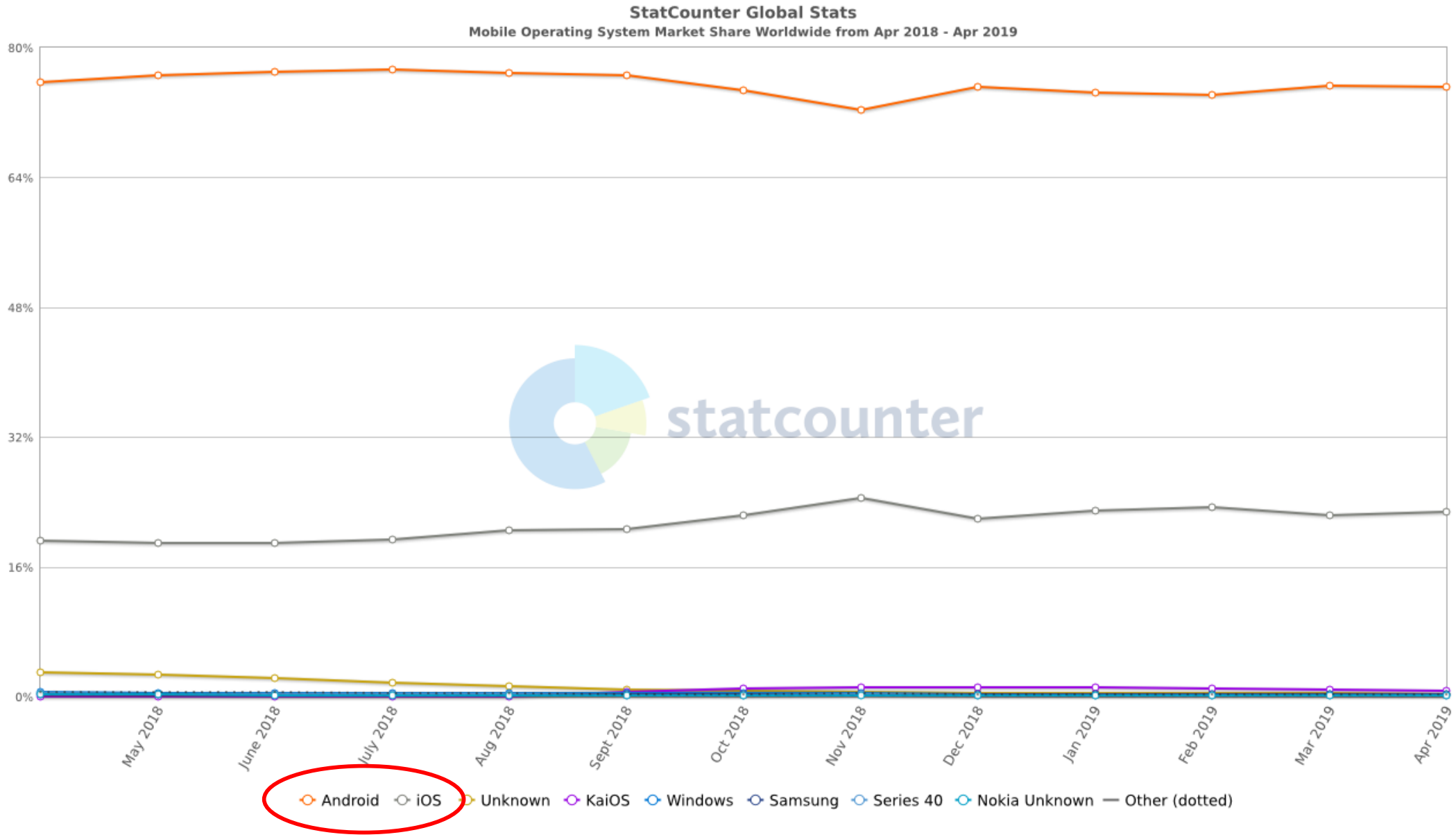
Quale smartphone attaccare? (3)

Cellulari (OS): quota di mercato mondiale - storia



Quali SO analizziamo? (1)

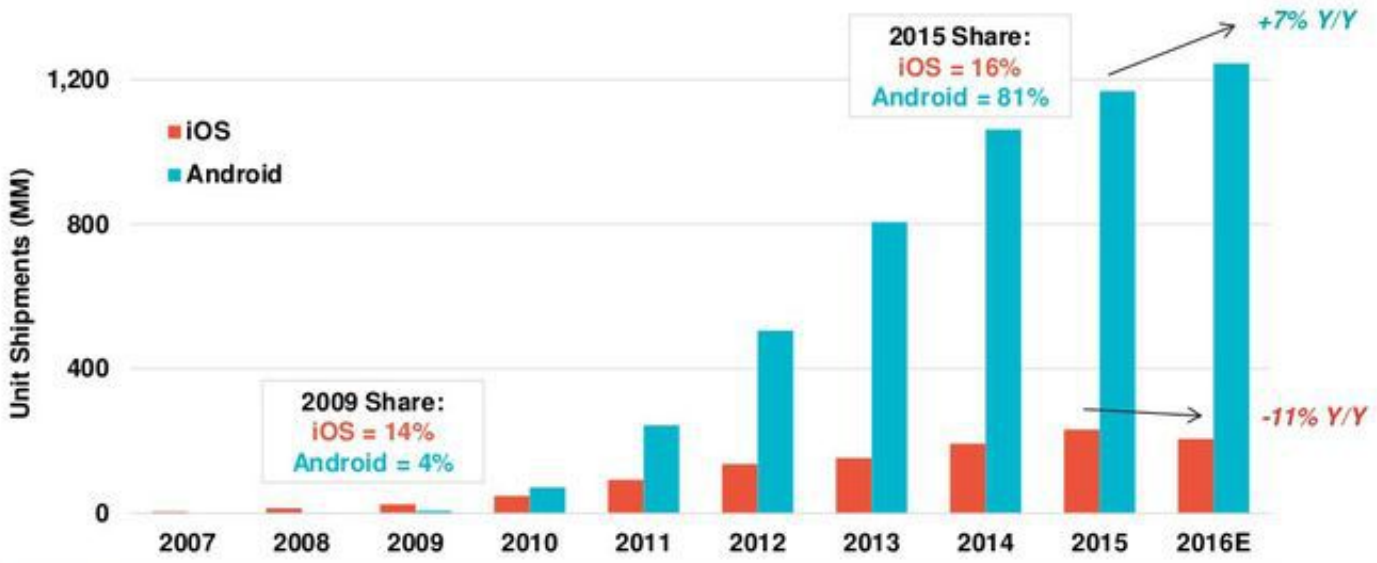
Quali SO analizziamo? (2)



Quali SO analizziamo? (3)

Android Smartphone Share Gains Continue vs. iOS... Android ASP Declines Continue...Delta to iOS @ ~3x

Smartphone Unit Shipments, iOS vs. Android, Global, 2007 – 2016E



iOS ASP ($)	$594	$621	$623	$703	$712	$686	$669	$680	$717	$651
Y/Y Growth	–	*4%*	*0%*	*13%*	*1%*	*-4%*	*-2%*	*2%*	*5%*	*-9%*
Android ASP	–	$403	$435	$441	$380	$318	$272	$237	$216	$208
Y/Y Growth	–	–	*8%*	*1%*	*-14%*	*-16%*	*-15%*	*-13%*	*-8%*	*-4%*

@KPCB Source: Morgan Stanley Research, 5/16.

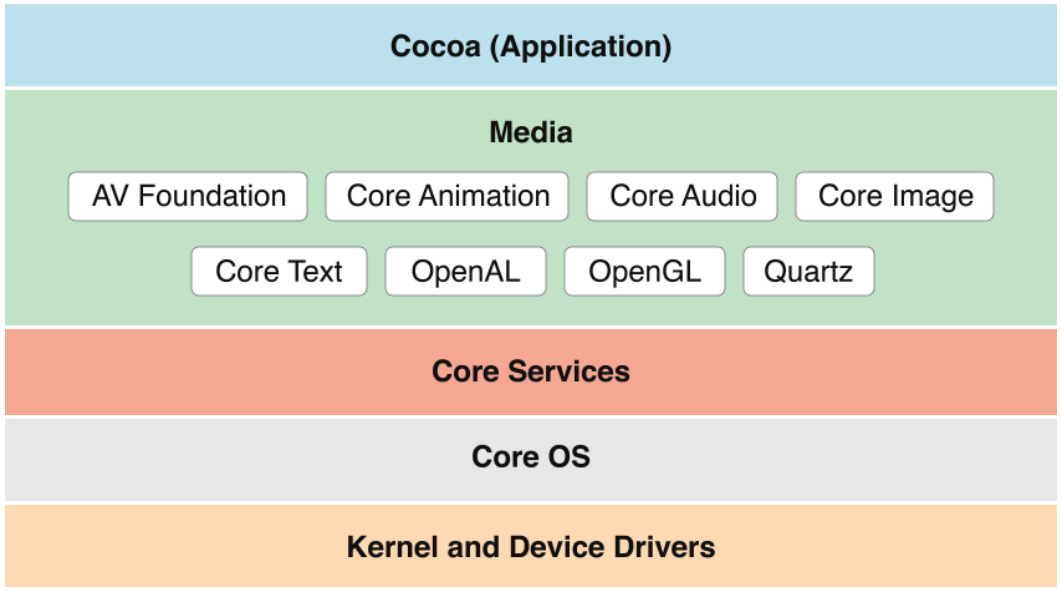
Android vs iOS

Due mondi diversi, con una filosofia di base diversa:

- *iOS* disponibile solo in smartphone (o dispositivi) Apple: i costi del top di gamma di Apple (e non solo) non sono certo accessibili a tutti
- *Android*, open source, frutto di una strategia diversa portata avanti da Google, che ha stretto accordi di carattere commerciale con svariati brand (in special modo Samsung) e con diverse fasce di prezzo, il che ha portato ad una diffusione su più larga scala

Apple iOS

Piattaforma iOS



- **Kernel**: basato su un Mach kernel come Mac OS X
- **Core OS** e **Core Services**: APIs verso hw, network, ecc.
  - Include SQLite, POSIX thread, UNIX socket
- **Media layer**: supporta grafica 2D e 3D, audio, video
- **Cocoa Touch**: responsabile dell'aspetto e della responsiveness alle azioni degli utenti, comprende:
  - **Foundation framework**: supporto per internationalization, object persistence, file management, network operations, and XML processing
  - **UIKit**: per lo sviluppo dell'interfaccia utente delle applicazioni
- Implementato in C e Objective-C

Sviluppo di Applicazioni iOS

- Le app sono sviluppate in Objective-C usando l'Apple Software Development Kit (SDK)
- Modello di gestione degli eventi basato su *touch events*
- I servizi di base usati da tutte le applicazioni sono definiti in:
  - Foundation framework: classi in Objective-C
  - UIKit framework: offre componenti HTML, CSS e JS per gestire l'interfaccia utente

Accesso sicuro al dispositivo (1)

Autenticazione/Sblocco dispositivo

- Codice di accesso
  - Necessario per configurare le altre due opzioni
  - Può essere di 4, 6, o indefiniti digit
- TouchID
- FaceID

- Dati biometrici:
  - MAI salvati su iCloud, iTunes,
  - MAI spediti a Apple
  - MAI inseriti nei backup

Accesso sicuro al dispositivo (2)

Autenticazione/Sblocco dispositivo

- Ci sono dei casi in cui viene obbligatoriamente richiesto il codice di accesso
  - dispositivo appena acceso
  - dispositivo inutilizzato (non sbloccato) per almento 48 ore
  - Non è stato utilizzato il codice per l'accesso per almeno 156 ore (6 giorni e mezzo) o FaceID nelle ultime 4 ore
  - il dispositivo ha ricevuto da remoto una richiesta di blocco
  - dopo 5 tentativi di riconoscimento falliti tramite TouchID/FaceID

Accesso sicuro al dispositivo (3)

Cosa succede se inserisco codici errati?

- Il modulo *Secure Enclave* applica un ritardo sempre più lungo dopo l'inserimento di un codice errato:

Ritardi tra i tentativi di inserimento del codice

Tentativi	Ritardo forzato
1–4	Nessuno
5	1 minuto
6	5 minuti
7–8	15 minuti
9	1 ora

- Quindi un attacco a forza bruta non è fattibile

System Security (0)

Secure Enclave

- ***Coprocessore*** che usa memoria cifrata e include un ***generatore di numeri casuali HW***
- Offre tutte le primitive crittografiche necessarie al modulo di Data Protection
- Si occupa anche dell'autenticazione tramite Touch ID e Face ID
  - Durante la fabbricazione viene assegnata una chiave simmetrica condivisa a Secure Enclave e a ciascun sensore

System Security (0)

Crypto Engine AES-256

- Molto efficiente
- Possiede due chiavi «fuse» in fase di fabbricazione, non leggibili, si vede solo il risultato dopo la cifratura/decifratura
  - Chiave UID, chiave a 256 bit, univoca per dispositivo
  - Chiave GID, chiave a 256 bit, comune a tutti i processori di una classe di dispositivi

System Security (1)

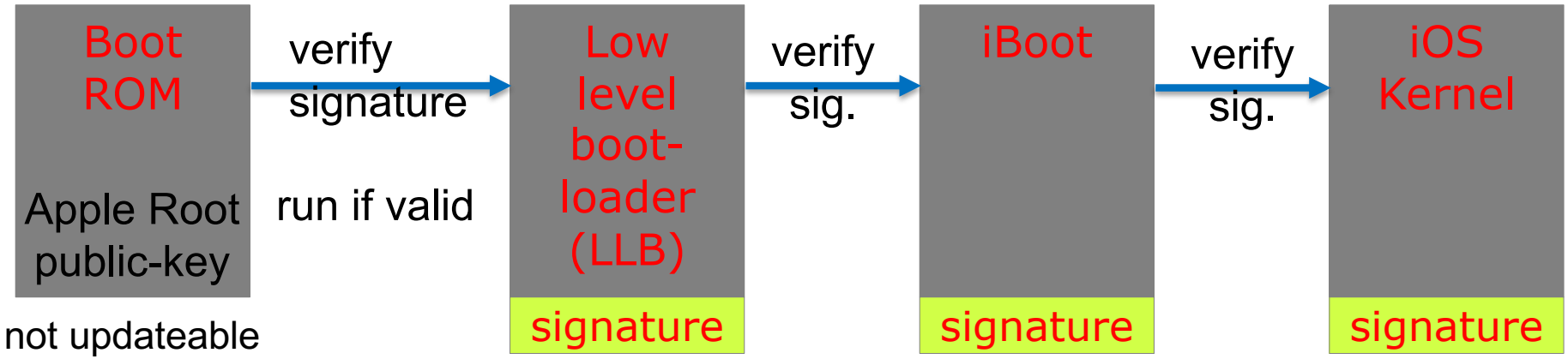
Procedura di avvio sicuro:

- I singoli componenti sono firmati da Apple (bootloader, kernel, estensioni del kernel e firmware) e vanno a formare una ***secure boot chain***
- La radice della catena è la ***Boot ROM***, che contiene la ***chiave pubblica Apple Root CA*** inserita nel chip durante la fabbricazione
- La chiave viene usata per verificare la firma del Boot ROM prima che venga caricato
  - Se passa la verifica, lo esegue e poi passa alla verifica del componente successivo
  - Se NON passa la verifica, va in DFU mode e presenta la scritta «collega a iTunes» per ripristinare le condizioni di fabbrica

System Security (1)

Procedura di avvio sicuro:

- I singoli componenti sono firmati da Apple (bootloader, kernel, estensioni del kernel e firmware) e vanno a formare una ***secure boot chain***



- La radice della catena è la ***Boot ROM***, che contiene la ***chiave pubblica Apple Root CA*** inserita nel chip durante la fabbricazione
- La chiave viene usata per verificare la firma del Boot ROM prima che venga caricato

System Security (2)

Rilascio sicuro di aggiornamenti

- Gli aggiornamenti vengono resi disponibili nello stesso momento per tutti i dispositivi
- L'utente riceve la notifica sul dispositivo
- Necessario che sia installato il sistema operativo originario, che richiede la firma di Apple
  - Viene mandata
    - una lista di "criptographic measurements" che vengono controllati da Apple
    - l'ECID (ID univoco del dispositivo)
    - un ***nonce***, per evitare replay attack
- Sono disponibili:
  - Aggiornamenti OTA (over the air)
  - Tramite iTunes

Sicurezza delle App (1)

Garanzia rispetto all'origine delle App:

- Per poter sviluppare App per iOS il programmatore deve **registrarsi** (vengono effettuate delle verifiche) presso Apple
- Le App vengono **controllate e validate** (nel giro di un paio di settimane)
  - Manualmente e in modo automatico
  - Controllano assenza di violazioni di sicurezza e privatezza
- Le app devono venire **firmate** da un certificato emesso da Apple (mandatory code signing) e devono contenere l'identità dello sviluppatore

Sicurezza delle App (2)

Esecuzione sicura delle App (Runtime protection):

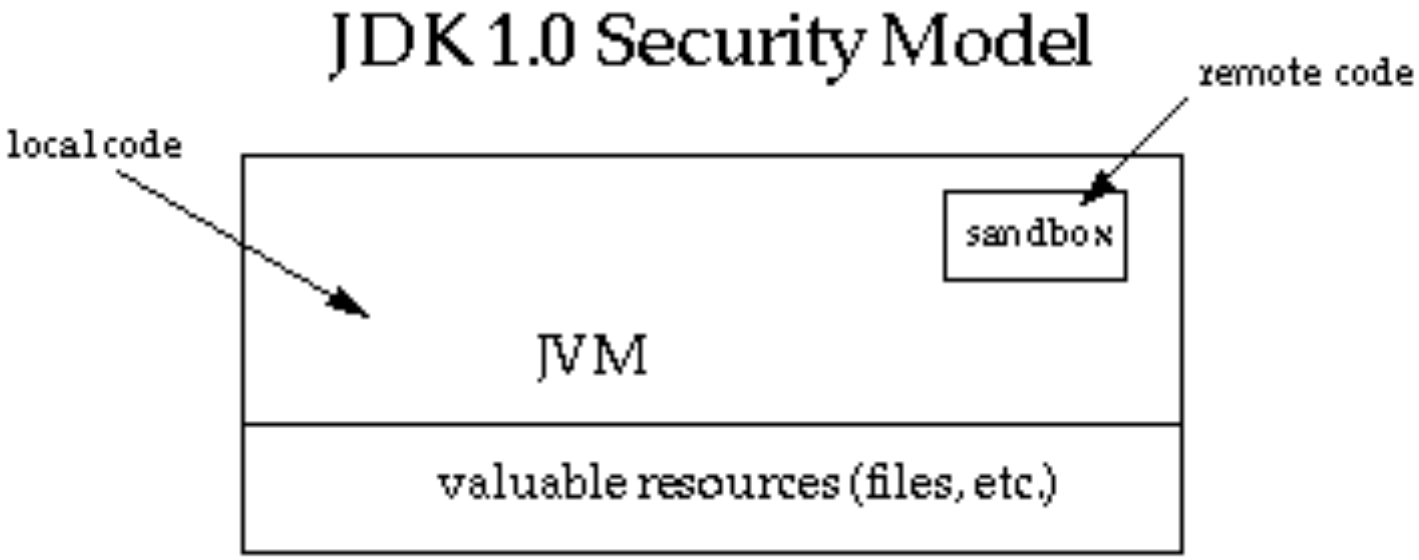
- Le risorse del sistema e il kernel sono separati (e protetti!) dalle applicazioni utente
- App "sandbox" nega l'accesso ai dati di altre app
- La comunicazione tra app avviene solo mediante iOS API

Sandbox

- Termine inglese con cui si indica il recinto della sabbia destinato ai giochi dei bambini dove possono giocare “protetti”
- In ambito informatico:
  - identifica un ambiente di test spesso slegato dal normale flusso di ambienti predisposti per lo sviluppo e il test delle applicazioni
  - il modello originale progettato per confinare codice potenzialmente dannoso in un ambiente isolato e fortemente restrittivo

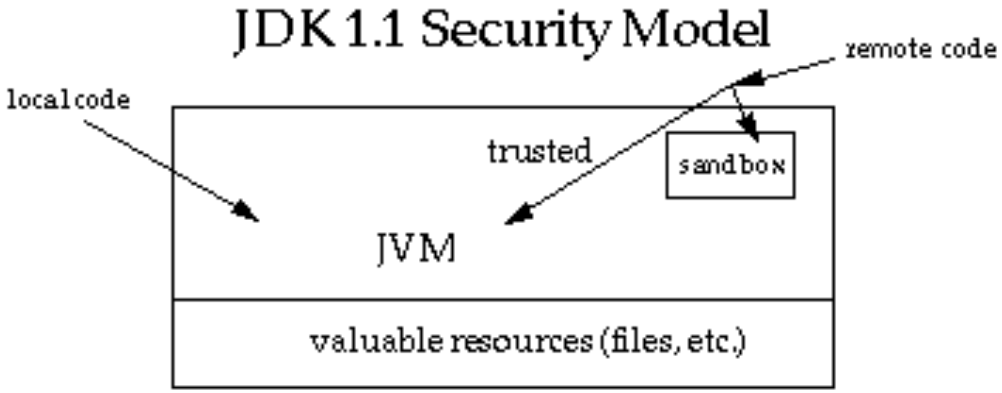
Java Sandbox (1)

- Java, sin dalla sua nascita, fu fortemente orientato alla rete
  - applet: codice direttamente scaricato da remoto ed eseguito, esponendo il client a notevoli problematiche di sicurezza
- **Java Sandbox v1**:
  - distingue solo tra *codice locale* (con accesso completo a tutte le risorse, anche critiche –filesystem-, del sistema) e *codice remoto* (con limitato accesso alle risorse, mediato dalla sandbox stessa)

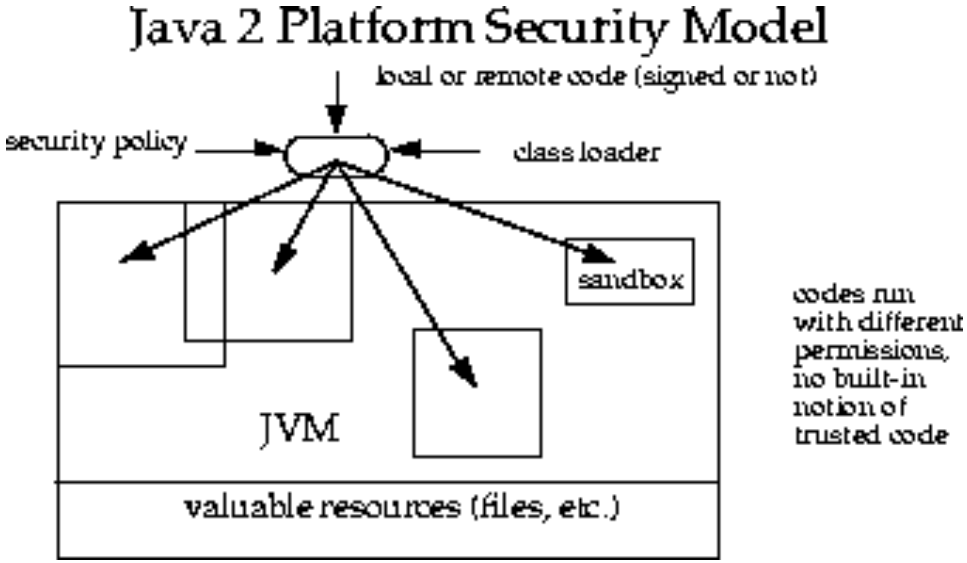


Java Sandbox (2)

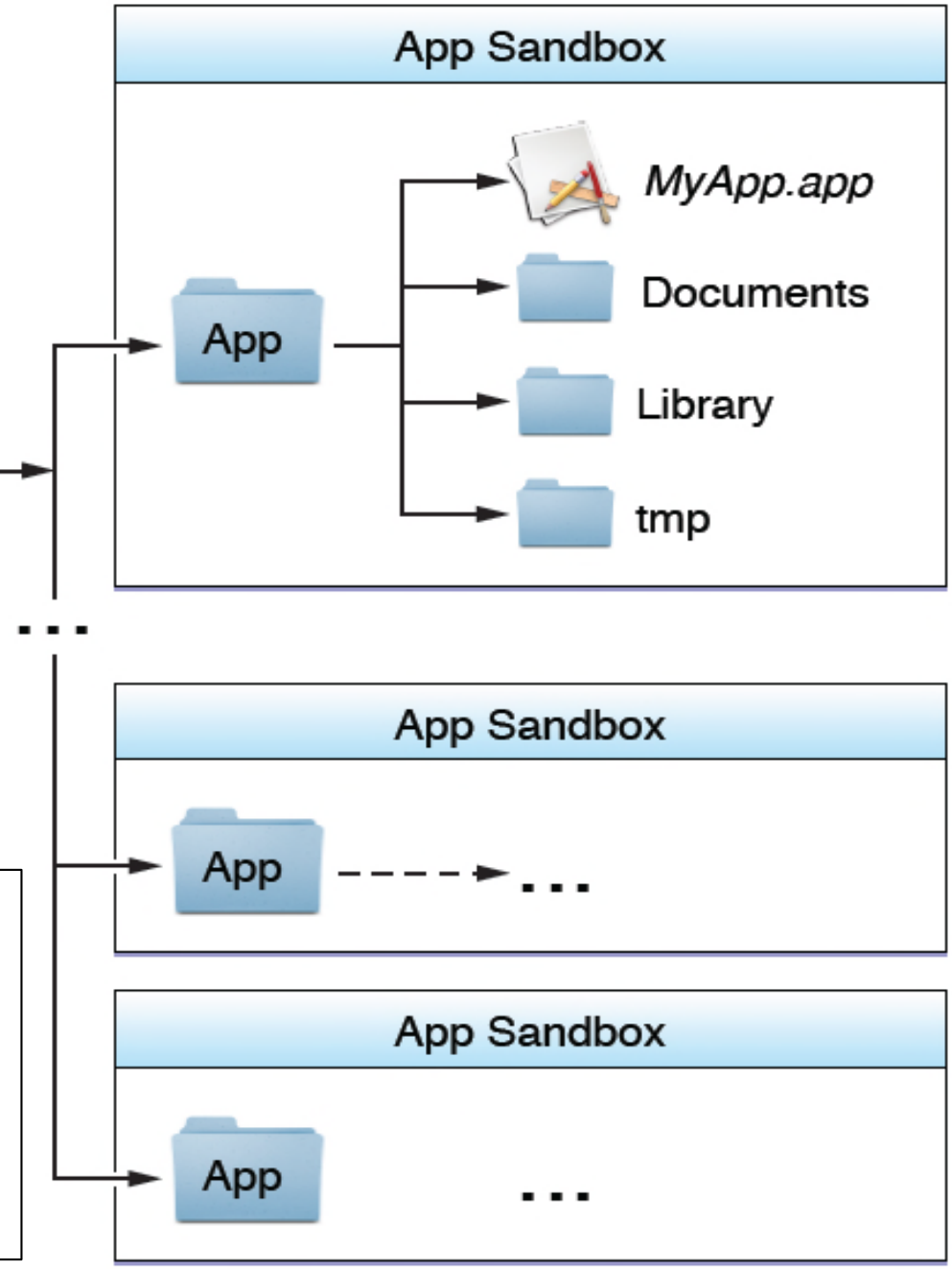
- **Java Sandbox v2**:



- **Java Sandbox v3**:



iOS Sandbox



- Limita l'accesso delle app a file, preferenze, rete e altre risorse
- Ciascuna app ha la sua cartella sandbox
- **Stessi privilegi per ciascuna app**
- Limita le conseguenze degli attacchi

Sicurezza delle App (3)

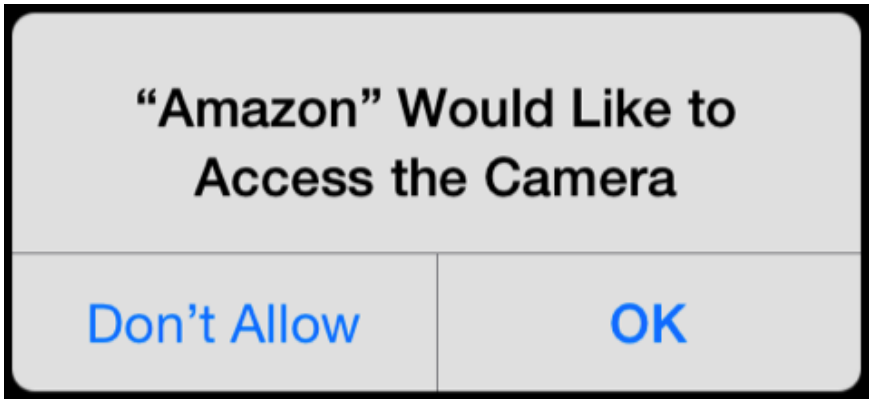
Permessi delle App: (1)

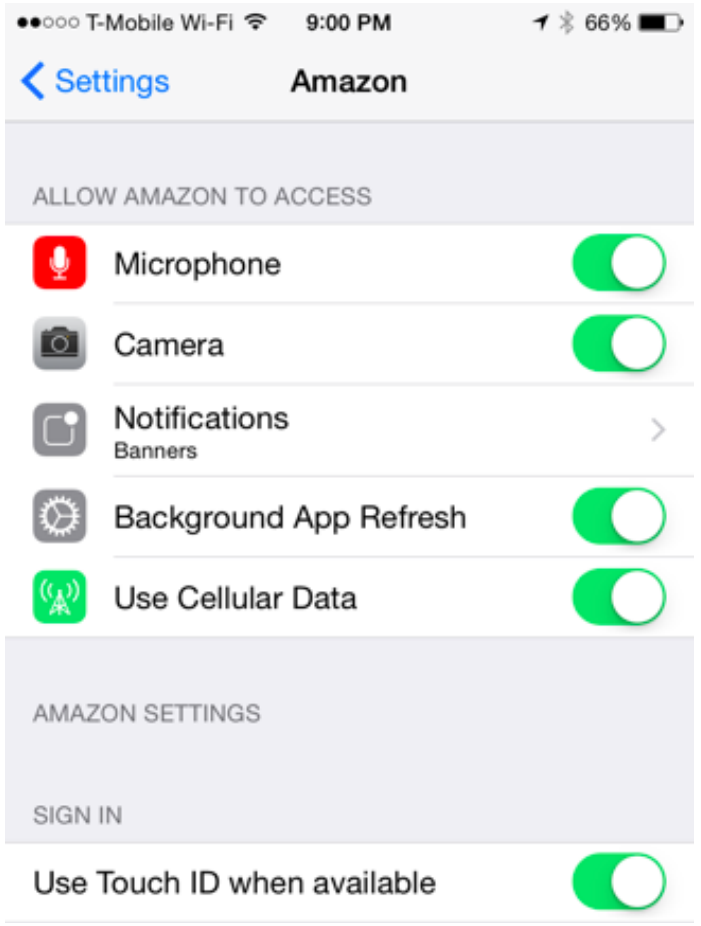
- Le App hanno tutte gli stessi permessi di default
- Viene chiesto all'utente, ove necessario, se concedere ulteriori permessi, come:
  - Accesso a dati relativi alla posizione
  - Ricevere notifiche push
  - Inviare sms
  - Far partire una chiamata in uscita

Sicurezza delle App (4)

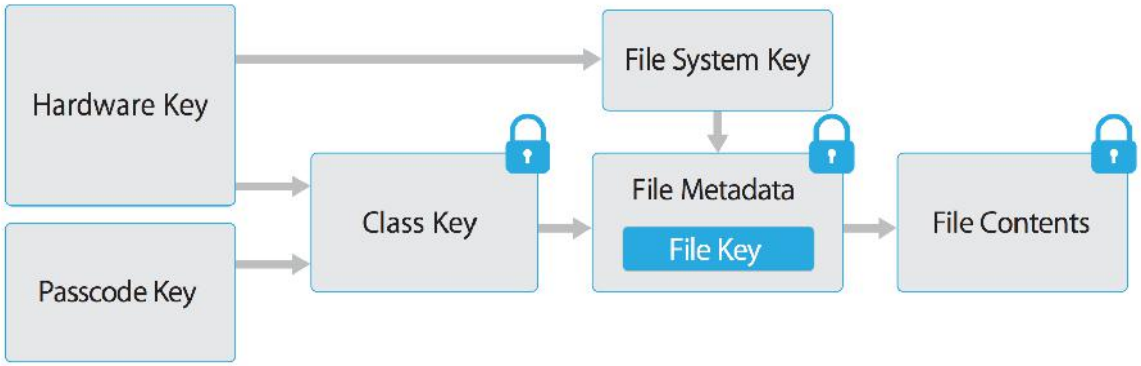
Permessi delle App: (2)

- L’utente li può gestire facilmente





Data Security (1)



Passcode key:

- Chiave derivate dal codice di sicurezza (hash del codice e dell'ID del dispositivo)

Hw key:

- Chiave inserita durante la fabbricazione

Chiave per classe

- Un file appartiene ad una classe

Chiave per file o per extent

- Nel secondo caso posso cifrare parti di file con chiavi diverse

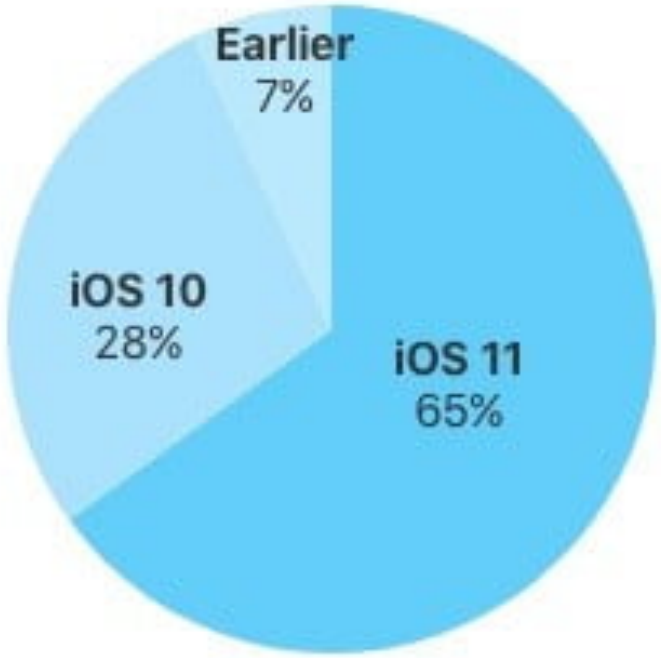
Apple iOS: distribuzione (1)

iOS	Ultima versione	Data di pubblicazione	
		Prima versione	Ultima versione
iPhone OS 1	1.1.5	29 giugno 2007	15 luglio 2008
iPhone OS 2	2.2.1	11 luglio 2008	27 gennaio 2009
iPhone OS 3	3.2.2	17 giugno 2009	11 agosto 2010
iOS 4	4.3.5	21 giugno 2010	25 luglio 2011
iOS 5	5.1.1	12 ottobre 2011	7 maggio 2012
iOS 6	6.1.6	19 settembre 2012	21 febbraio 2014
iOS 7	7.1.2	18 settembre 2013	30 giugno 2014
iOS 8	8.4.1	17 settembre 2014	13 agosto 2015
iOS 9	9.3.5	16 settembre 2015	25 agosto 2016
iOS 10	10.3.3	13 settembre 2016	19 luglio 2017
iOS 11	11.3.1	19 settembre 2017	24 aprile 2018

Versione		
Numero	Data di distribuzione	iPhone
3.1.3	2 febbraio 2010	2G
4.2.1	25 novembre 2011	3G
5.1.1	7 maggio 2012	
6.1.6	21 febbraio 2014	3GS
7.1.2	30 giugno 2014	4
9.3.5	25 agosto 2016	4S
10.3.3	19 luglio 2017	5 · 5c
11.3.1	24 aprile 2018	5s · 6 · 6 Plus · 6s · 6s Plus · SE · 7 · 7 Plus · 8 · 8 Plus · X

Apple iOS: distribuzione (2)

65% of devices are using iOS 11.



As measured by the App Store on January 18, 2018.

iOS	Ultima versione	Data di pubblicazione	
		Prima versione	Ultima versione
iPhone OS 1	1.1.5	29 giugno 2007	15 luglio 2008
iPhone OS 2	2.2.1	11 luglio 2008	27 gennaio 2009
iPhone OS 3	3.2.2	17 giugno 2009	11 agosto 2010
iOS 4	4.3.5	21 giugno 2010	25 luglio 2011
iOS 5	5.1.1	12 ottobre 2011	7 maggio 2012
iOS 6	6.1.6	19 settembre 2012	21 febbraio 2014
iOS 7	7.1.2	18 settembre 2013	30 giugno 2014
iOS 8	8.4.1	17 settembre 2014	13 agosto 2015
iOS 9	9.3.5	16 settembre 2015	25 agosto 2016
iOS 10	10.3.3	13 settembre 2016	19 luglio 2017
iOS 11	11.3.1	19 settembre 2017	24 aprile 2018

Mobile threats? (3) – Contromisure?

Aspetti di sicurezza che deve considerare una piattaforma mobile:

- Accesso sicuro al dispositivo
- Sicurezza del sistema
- Sicurezza delle App
- Sicurezza dei dati (sul cellulare e su cloud)

Android

Un po' di storia... (1)

- Ottobre 2003: Andy Rubin (ex WebTV e Danger Inc. – una compagnia che aveva sviluppato un suo smartphone e un suo OS scritto in Java) fonda Android Inc. insieme ad alcuni amici/colleghi
- 17 Agosto 2005: Google acquisisce Android Inc.
- 5 novembre 2007: nel blog ufficiale di Google, Rubin annuncia:
  - *Android is the first truly open and comprehensive platform for mobile devices*
  - *We have developed Android in cooperation with the Open Handset Alliance (OHA)*
    - Open Handset Alliance, un gruppo all'inizio di circa 30 carrier, mobile device e component manufacturers and sw vendors (T-Mobiel, Motorola, Qualcomm, HTC)

Un po' di storia... (2)

- 12 Novembre 2007: viene rilasciata la prima versione dell'*Android Software Development kit* (*SDK*)
- Gennaio-Aprile 2008: viene lanciata la prima Android Developer Challange
  - $ 1,000,000 per le App più innovative
- 28 Agosto 2008: viene lanciato *Android Market*
  - All'inizio non è previsto un metodo di pagamento delle App
- 22 Ottobre 2008: primo dispositivo che supporta Android lanciato sul mercato (T-Mobile G1 - HTC Dream)

Un po’ di storia… (3)

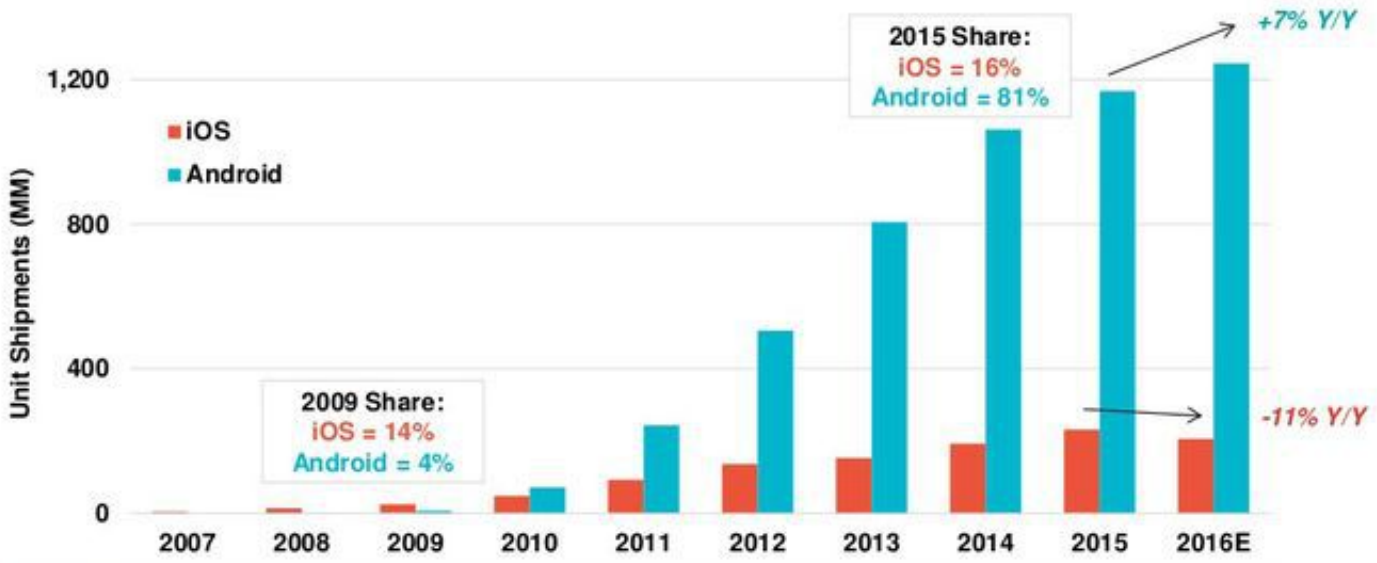
Versioni di Android:

- 1.5 ***Cupcake***
- 1.6 ***Donut***
- 2.0 ***Eclair***
- 2.2 ***Froyo***
- 2.3 ***Gingerbread***
- 3.0 ***Honeycomb***
- 4.0 ***Ice Cream Sandwich***
- 4.1 ***Jelly Bean***
- 4.4 ***KitKat***
- 5.0 ***Lollipop***
- 6.0 ***Marshmallow***
- 7.0 ***Nougat***
- 8.0 ***Oreo***
- 9.0 ***Pie***

Un po' di storia... (4)

Android Smartphone Share Gains Continue vs. iOS... Android ASP Declines Continue...Delta to iOS @ ~3x

Smartphone Unit Shipments, iOS vs. Android, Global, 2007 – 2016E

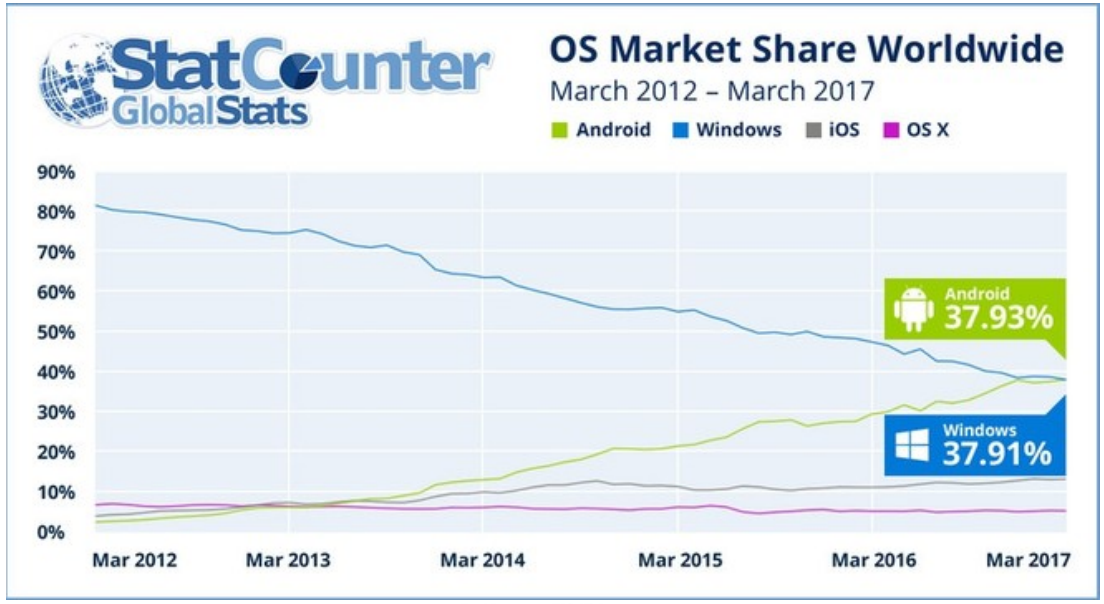


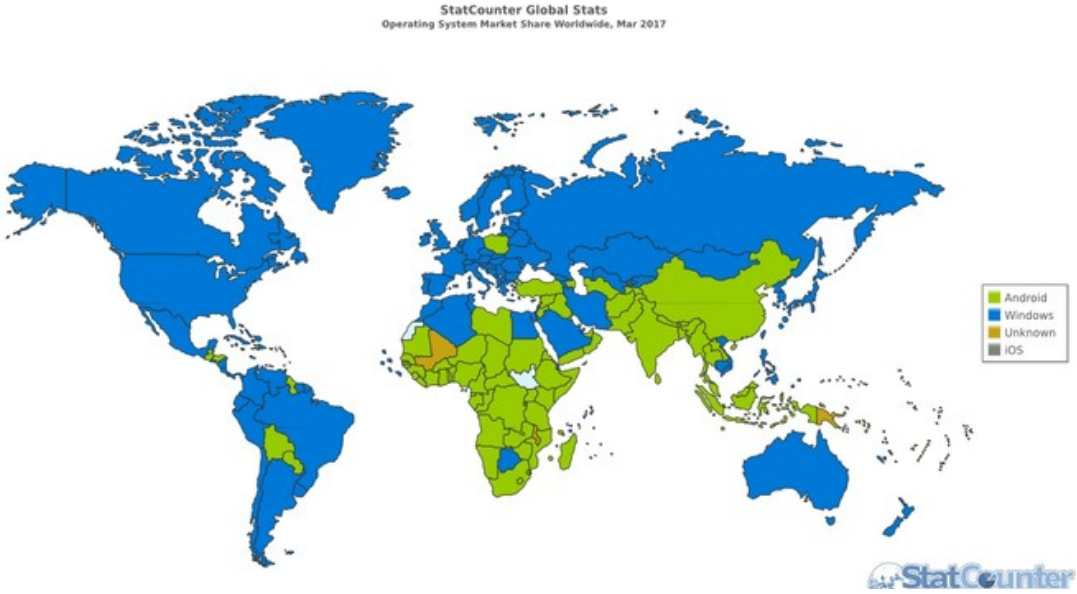
iOS ASP ($)	$594	$621	$623	$703	$712	$686	$669	$680	$717	$651
Y/Y Growth	–	*4%*	*0%*	*13%*	*1%*	*-4%*	*-2%*	*2%*	*5%*	*-9%*
Android ASP	–	$403	$435	$441	$380	$318	$272	$237	$216	$208
Y/Y Growth	–	–	*8%*	*1%*	*-14%*	*-16%*	*-15%*	*-13%*	*-8%*	*-4%*

@KPCB Source: Morgan Stanley Research, 5/16.

Un po’ di storia… (5)

Statistiche relative alla diffusione dei **sistemi operativi più usati per navigare in Internet**:

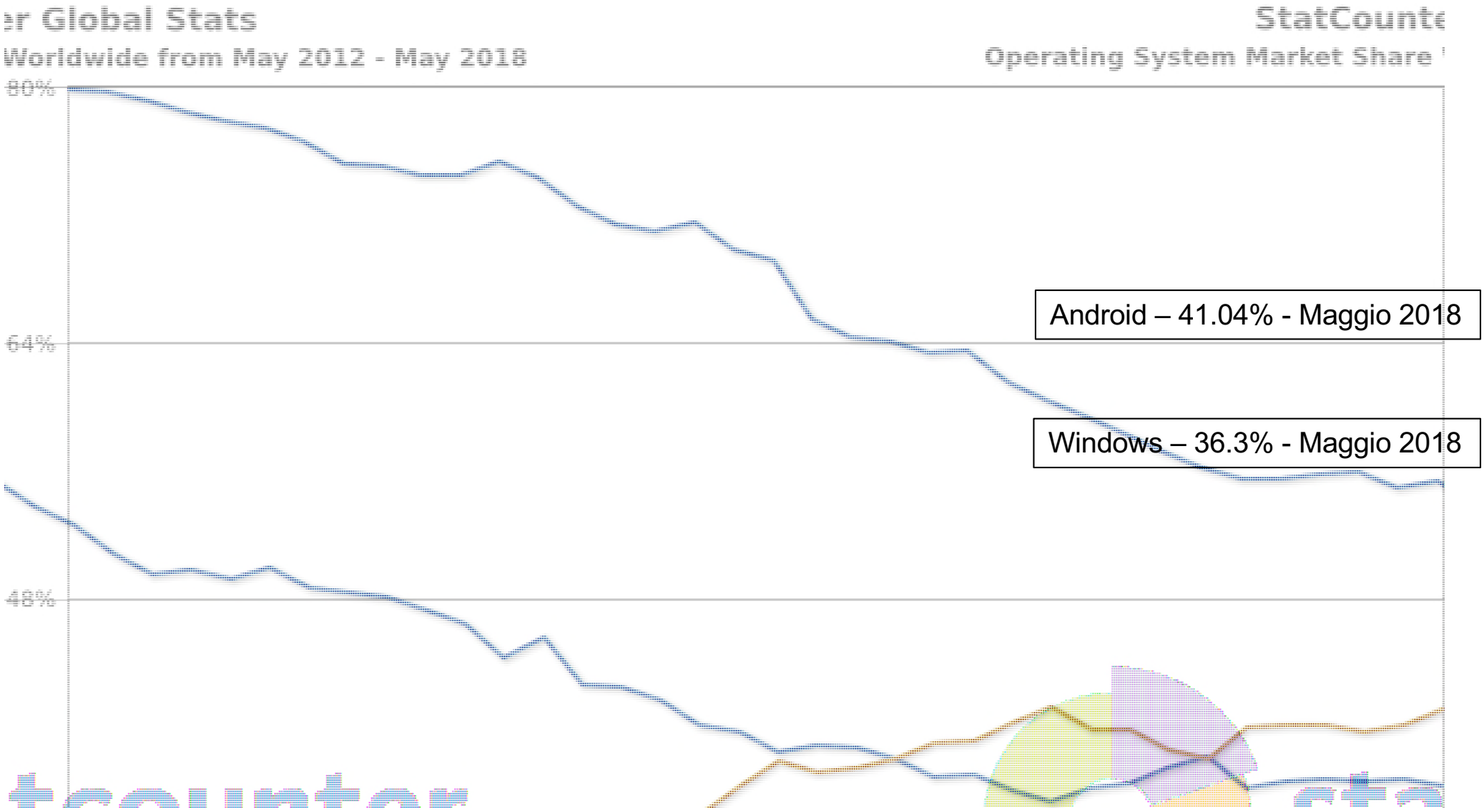




- Dati ottenuti da tracker Web installati su circa 2,5 milioni di siti
- Non viene calcolato il traffico generato dalle App
- Nell’immagine in basso: primati suddivisi per stato

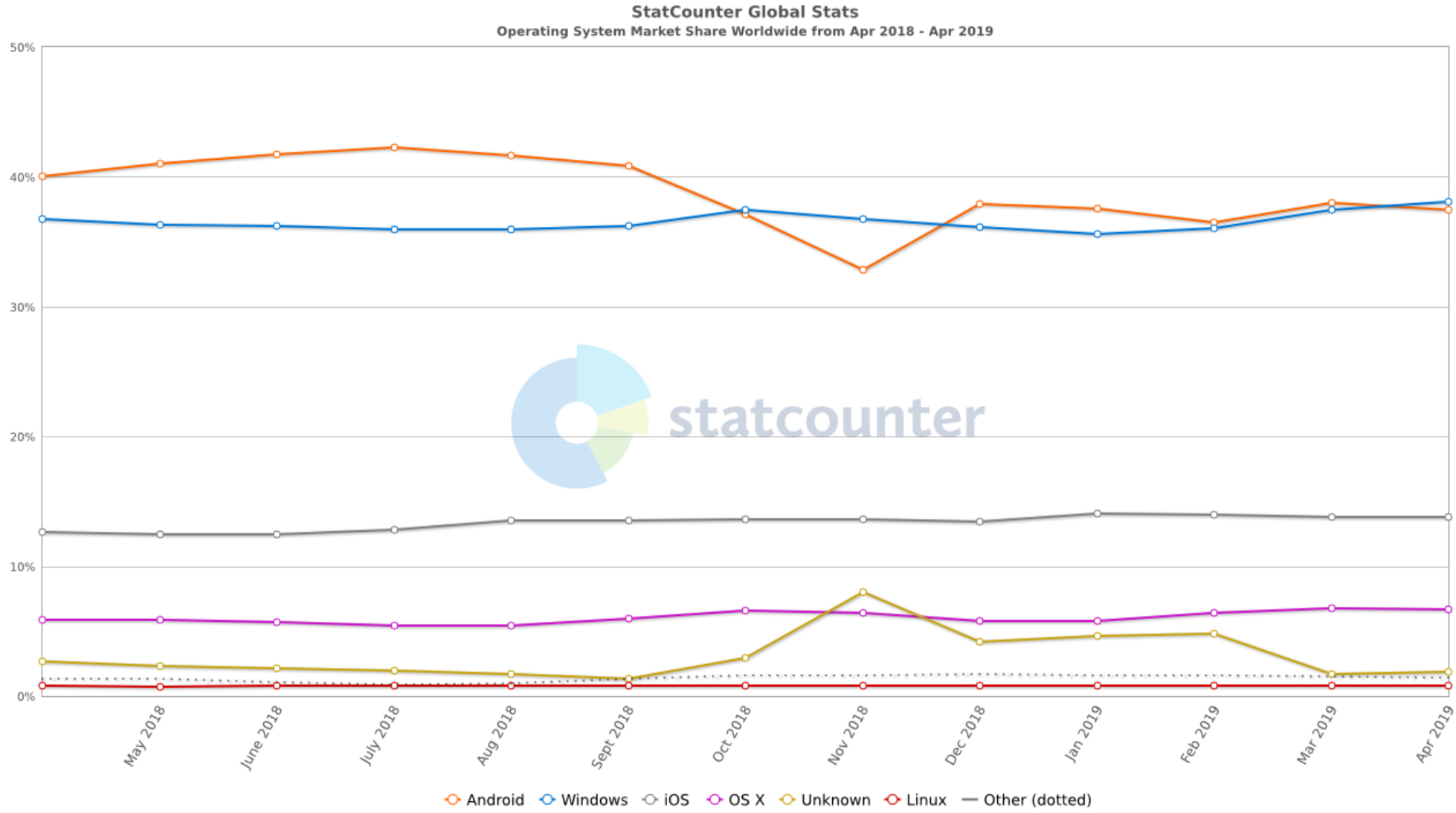
Un po’ di storia… (6)

Statistiche relative alla diffusione dei **sistemi operativi più usati per navigare in Internet**:



Un po’ di storia… (7)

Statistiche relative alla diffusione dei **sistemi operativi più usati per navigare in Internet**:



Un po’ di storia… (8)

Esistono interfacce utente specializzate:

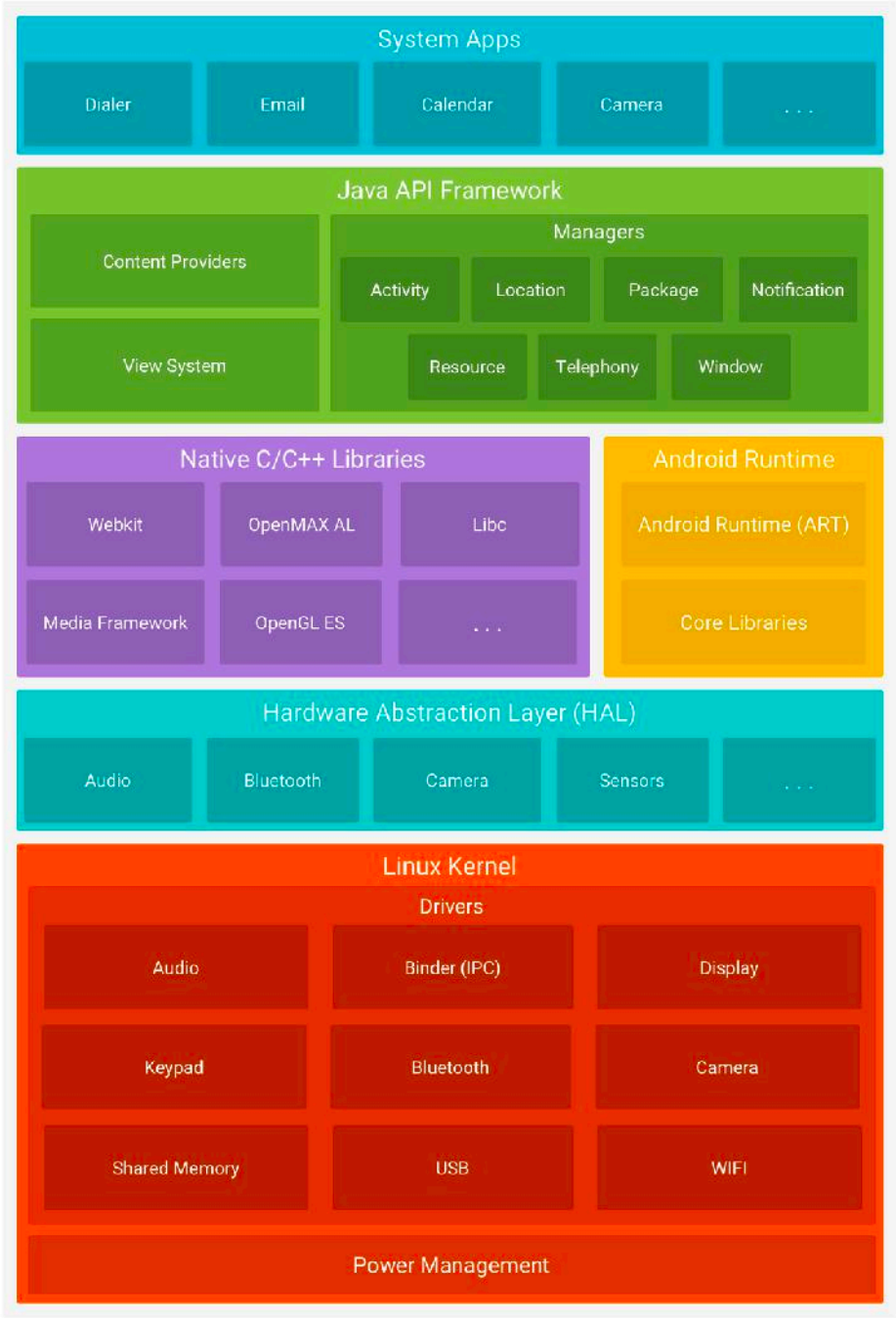
- **Wear OS**: per orologi da polso
- **Android TV**: per televisori
- **Android Auto**: per automobili
- **Android Things**: per oggetti intelligenti
- **Google Glass**: per occhiali

Un po’ di storia… (9)

Perché open source?

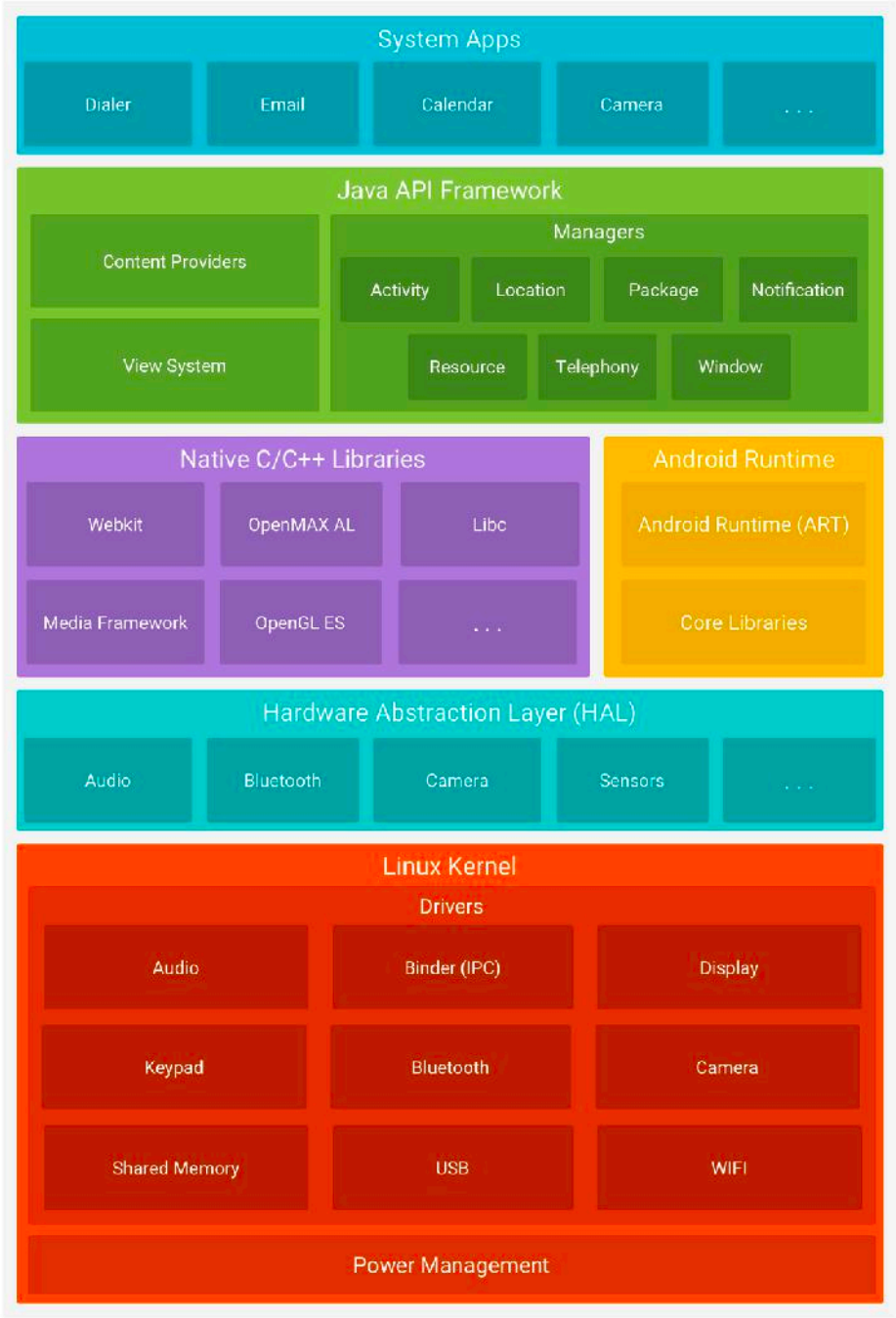
- Per il produttore di cellulari: in passato ogni azienda produceva il proprio SO, dopo l’uscita dell’iPhone difficile competere
  - con Android hanno continuato a produrre i propri dispositivi hardware usando come sistema operativo Android, dando vita all'enorme successo di questo OS
- **Per lo sviluppatore**: approccio unificato allo sviluppo di applicazioni
  - gli sviluppatori devono sviluppare soltanto per questo sistema e le loro applicazioni saranno in grado di funzionare su una molteplicità di dispositivi differenti, prodotti da svariate aziende (es. Samsung, LG, Huawei, Sony, ecc.).

Piattaforma Android



- **Kernel Linux**: offre le funzionalità di base, in particolare i *driver* a basso livello per i vari componenti HW di un dispositivo
- **HW Abstraction Layer (HAL)**: *interfaccia* rispetto ai singoli componenti HW (camera, Blootooth, ecc.)
- **Libraries** – *librerie* che contengono il codice che fornisce le funzionalità principali del SO
  - **WebKit**: funzionalità per la navigazione sul web attraverso il browser
- **Android Runtime** (dalla 5.0): implementazione "customizzata" di VM
  - Esegue file con formato ottimizzato (.dex)

Piattaforma Android

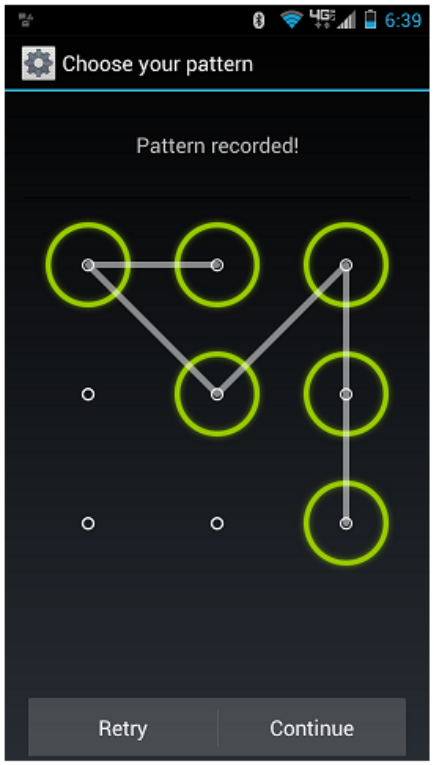


- **Java API Framework**: core da utilizzare per lo sviluppo di App
- **System Apps**: contiene le applicazioni, sia quelle incluse nel dispositivo, che quelle installate successivamente

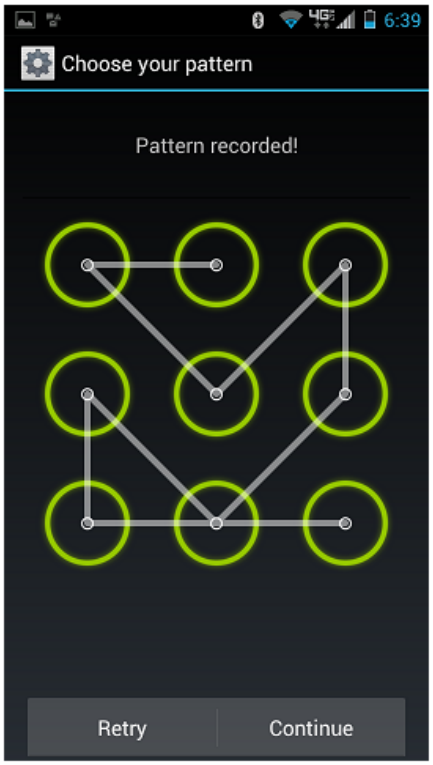
Accesso sicuro al dispositivo (1)

Meccanismo di Autenticazione/Sblocco dispositivo

- Codice di accesso
- Pattern
- Fingerprint (in alcuni dispositivi)



(a)



(b)

System Security (1)

Procedura di avvio sicuro:

- Android dalla versione 4.4 supporta il ***verified boot*** che avviene in due fasi:
  1. ***Secure booting***
     - La boot image `boot.img` contiene alla fine del blocco una firma
     - Quando l'immagine viene caricata il *bootloader* verifica la firma utilizzando una chiave che è memorizzata nel *secure keystore* del dispositivo (anche il produttore può inserire la sua chiave nel dispositivo)
     - A run-time il bootloader può essere in tre stati: *Locked, Verified, Unlocked*
     - In generale i dispositivi vengono venduti con lo stato *Locked,* quindi l'immagine non può venire cambiata o rimossa

System Security (2)

Procedura di avvio sicuro:

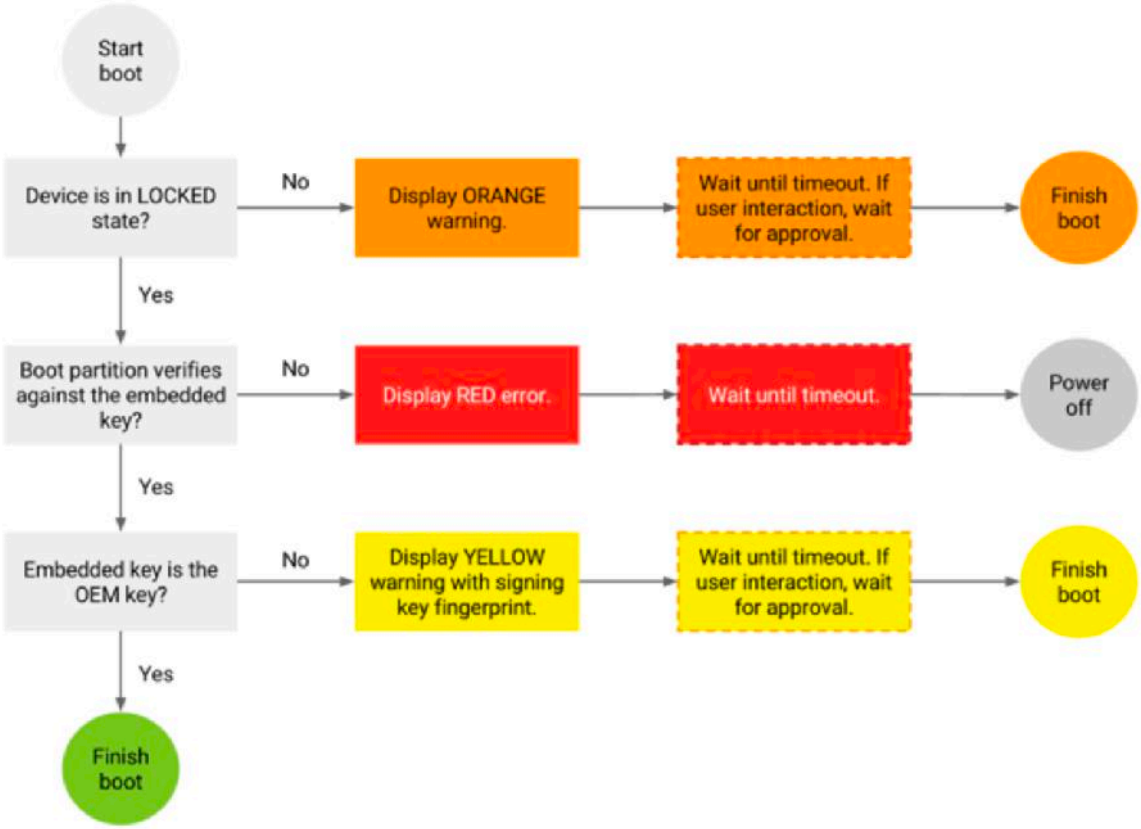
- Android dalla versione 4.4 supporta il ***verified boot*** che avviene in due fasi:
  2. ***System image verification***
     - Viene utilizzata una feature del kernel through device-mapper-verity (dm-verity) che verifica l'integrità dell'immagine caricata in fase di avvio

System Security (3)

Device state

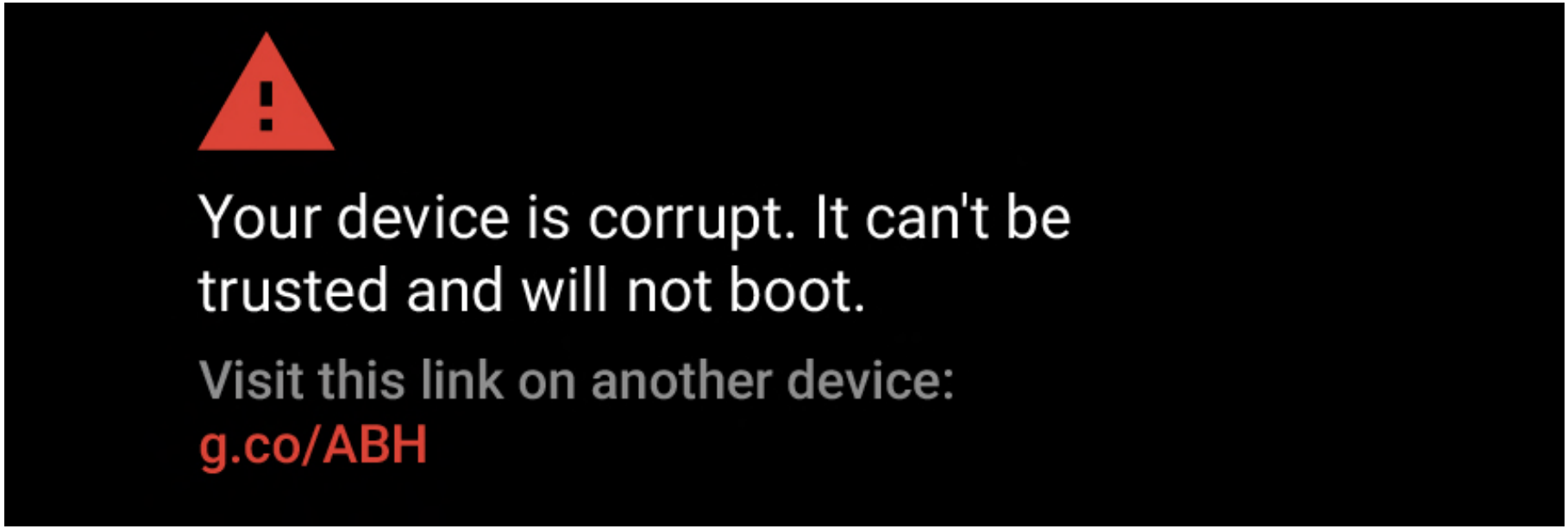
The possible device states and their relationship with the four verified boot states are:

1. LOCKED, indicating the device cannot be flashed. A LOCKED device boots into the GREEN, YELLOW, or RED states during any attempted boot.
2. UNLOCKED, indicating the device may be flashed freely and is not intended to be verified. An UNLOCKED device always boots to the ORANGE boot state.



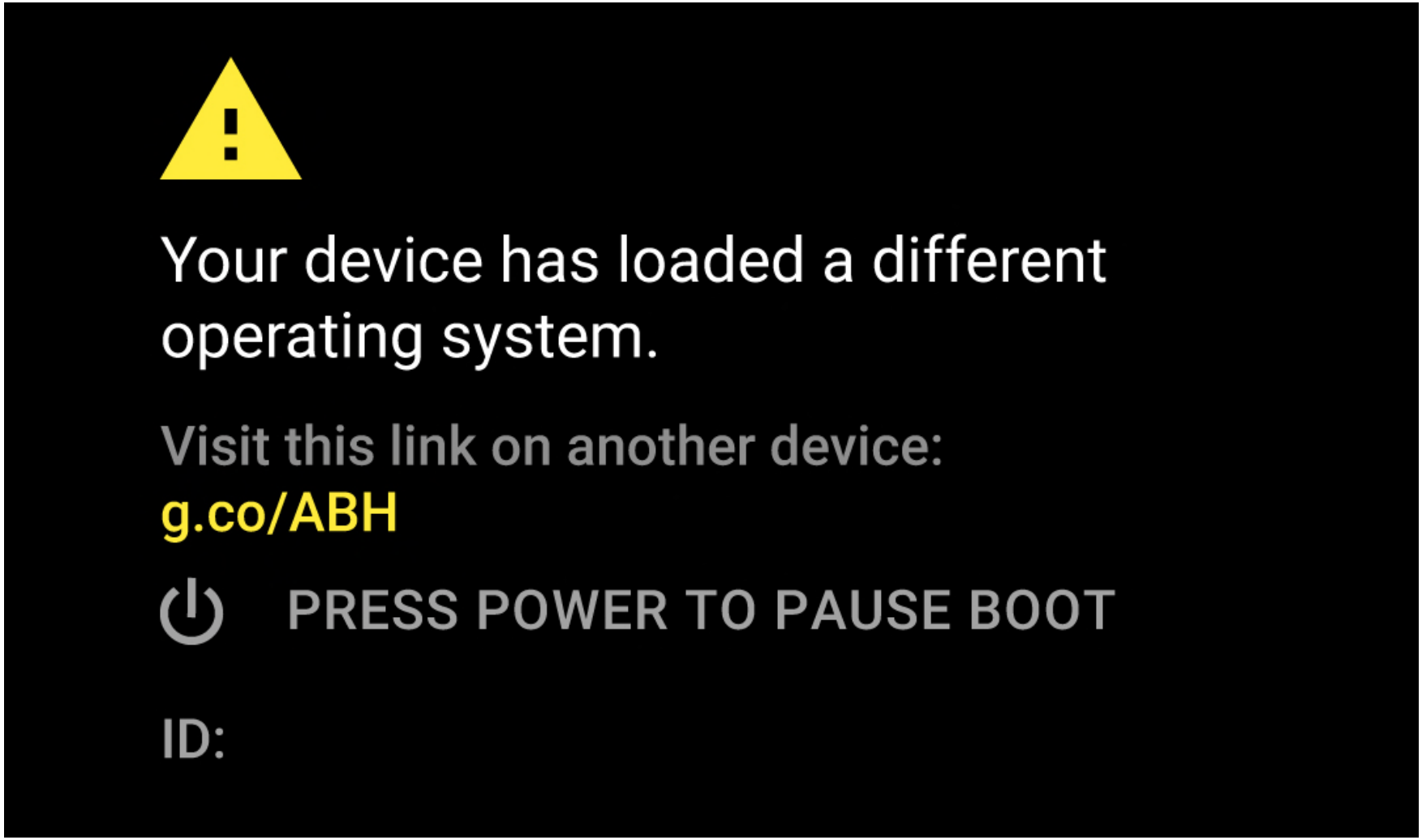
System Security (4)

Procedura di avvio sicuro:



System Security (5)

Procedura di avvio sicuro:



System Security (6)

Rilascio sicuro di aggiornamenti

Quali aggiornamenti?

- Le ultime versioni di Android **non** arrivano nello stesso momento su tutti i produttori
- Da quando Google rilascia l'aggiornamento del SO possono passare settimane e addirittura mesi prima che un produttore, per esempio Huawei, Samsung o LG, lo renda disponibile su tutti i propri telefoni
- Alle volte i produttori, specie nella fascia entry-level, non rilasciano aggiornamenti disponibili

Android: Frammentazione (1)

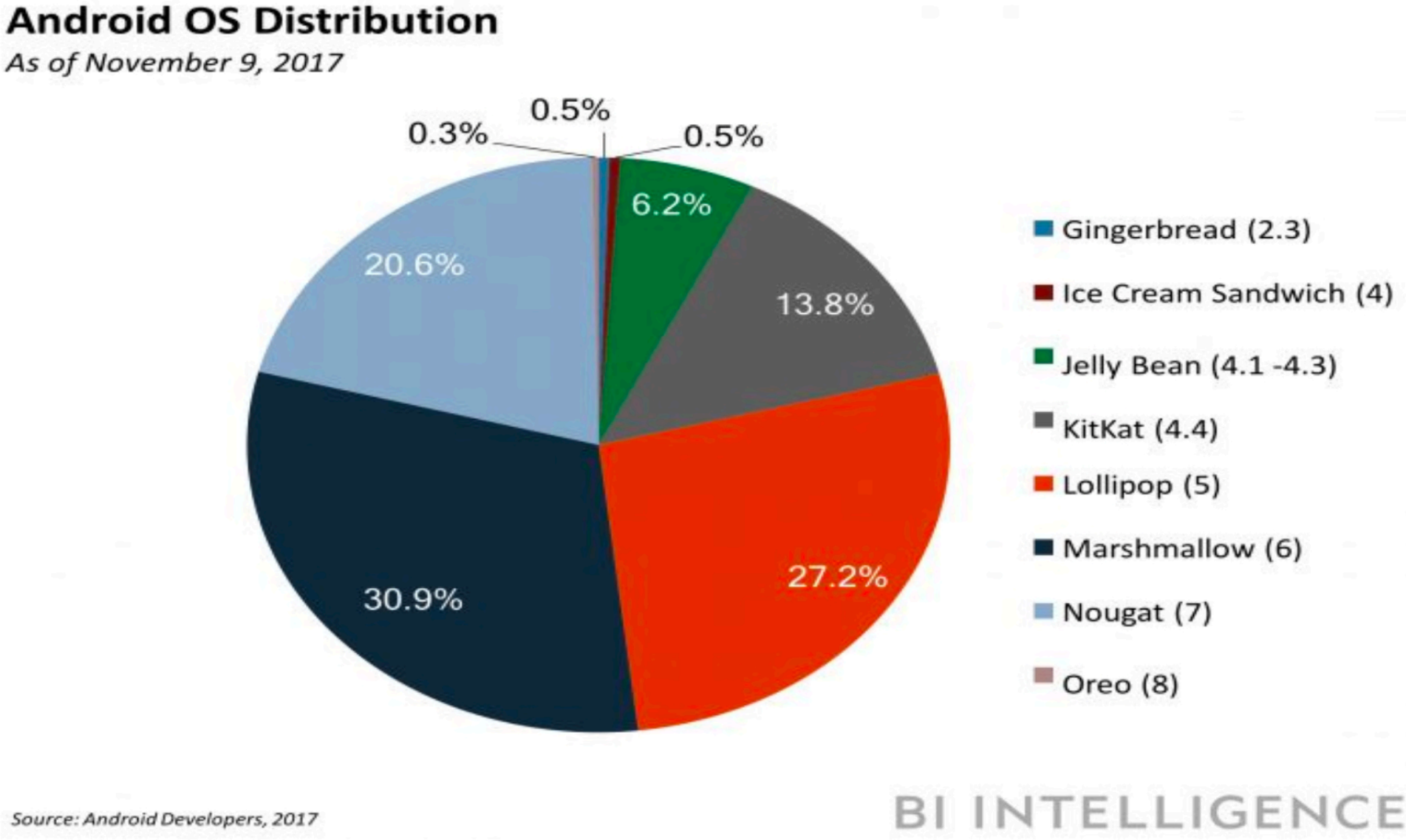
Frammentazione:

- Vecchie versioni del SO rimangono attive nei dispositivi

Perché un problema?

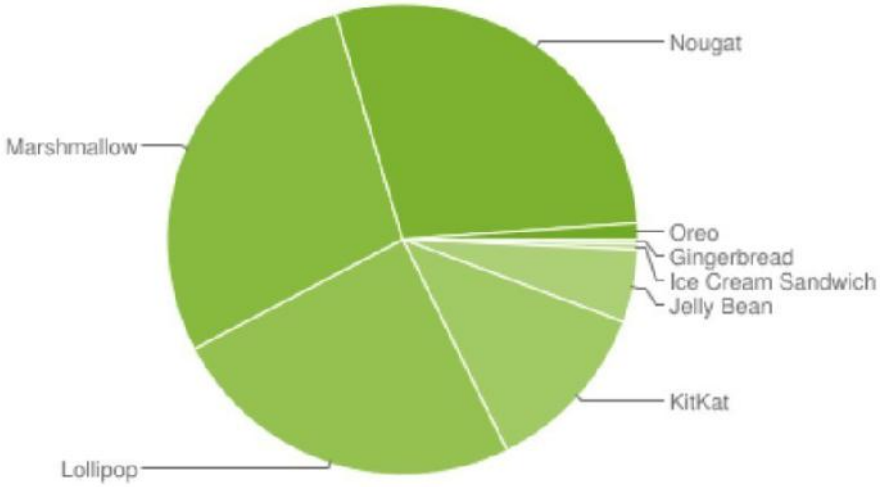
- Spesso le vecchie versioni hanno vulnerabilità (note!)
- Nei numeri: sembra che metà di tutti i dispositivi abbiano versioni vecchie di almeno 2 anni (da studio di Dan Luu)

Android: Frammentazione (2)



Android: Frammentazione (3)

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.4%
4.1.x	Jelly Bean	16	1.7%
4.2.x		17	2.6%
4.3		18	0.7%
4.4	KitKat	19	12.0%
5.0	Lollipop	21	5.4%
5.1		22	19.2%
6.0	Marshmallow	23	28.1%
7.0	Nougat	24	22.3%
7.1		25	6.2%
8.0	Oreo	26	0.8%
8.1		27	0.3%



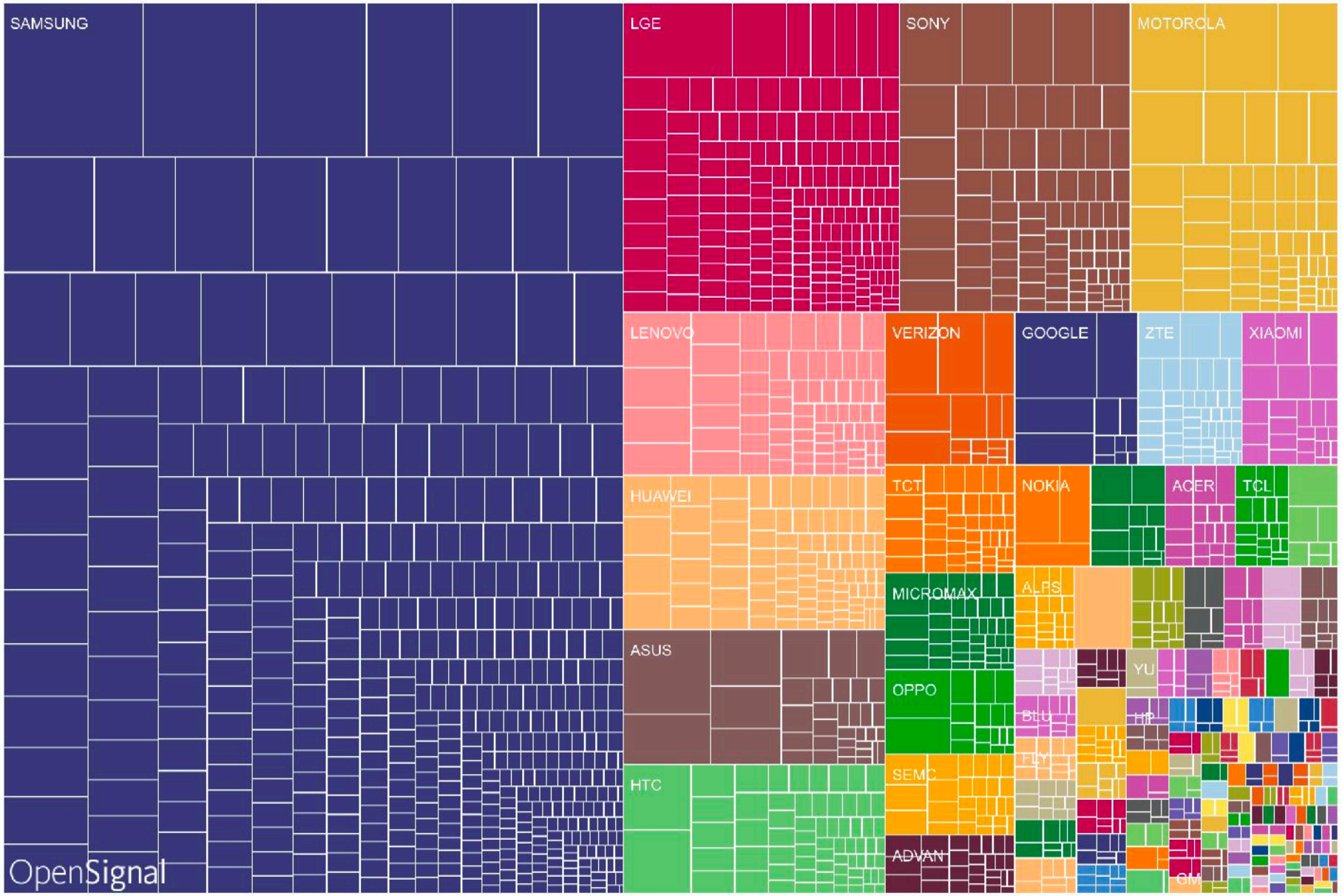
Data collected during a 7-day period ending on February 5, 2018.

Android: Frammentazione (4)

Device Fragmentation (da https://opensignal.com, Agosto 2015)

Android: Frammentazione (5)

Brand Fragmentation



Sicurezza delle App (1)

Garanzia rispetto all'origine delle App:

- Tutte le App devono essere firmate, ma in genere sono ***self-signed***
  - Viene fatto meno controllo rispetto all'identità di chi crea l'account
  - Non esiste una Google CA: Gooogle non ha il controllo delle firme sulle App
- ***Mercato aperto***:
  - Esiste un mercato ufficiale, una volta *AndroidMarket* ora *GooglePlay*, ma anche dei mercati alternativi
  - *Google Bouncer:* Analisi statica e dinamica delle App prima del caricamento sul mercato ufficiale:
    - Analisi statica alla ricerca di vulnerabilità, Trojan, malware, ecc.
    - Esecuzione dell'App in ambiente protetto
    - Analisi dei dati sottomessi dagli sviluppatori

Sicurezza delle App (2)

Application Signing

Application signing allows developers to identify the author of the application and to update their application without creating complicated interfaces and permissions. Every application that is run on the Android platform must be signed by the developer. Applications that attempt to install without being signed will be rejected by either Google Play or the package installer on the Android device.

On Google Play, application signing bridges the trust Google has with the developer and the trust the developer has with their application. Developers know their application is provided, unmodified, to the Android device; and developers can be held accountable for behavior of their application.

On Android, application signing is the first step to placing an application in its Application Sandbox. The signed application certificate defines which user ID is associated with which application; different applications run under different user IDs. Application signing ensures that one application cannot access any other application except through well-defined IPC.

When an application (APK file) is installed onto an Android device, the Package Manager verifies that the APK has been properly signed with the certificate included in that APK. If the certificate (or, more accurately, the public key in the certificate) matches the key used to sign any other APK on the device, the new APK has the option to specify in the manifest that it will share a UID with the other similarly-signed APKs.

Applications can be signed by a third-party (OEM, operator, alternative market) or self-signed. Android provides code signing using self-signed certificates that developers can generate without external assistance or permission. Applications do not have to be signed by a central authority. Android currently does not perform CA verification for application certificates.

Applications are also able to declare security permissions at the Signature protection level, restricting access only to applications signed with the same key while maintaining distinct UIDs and Application Sandboxes. A closer relationship with a shared Application Sandbox is allowed via the shared UID feature where two or more applications signed with same developer key can declare a shared UID in their manifest.

https://source.android.com/security/overview/app-security

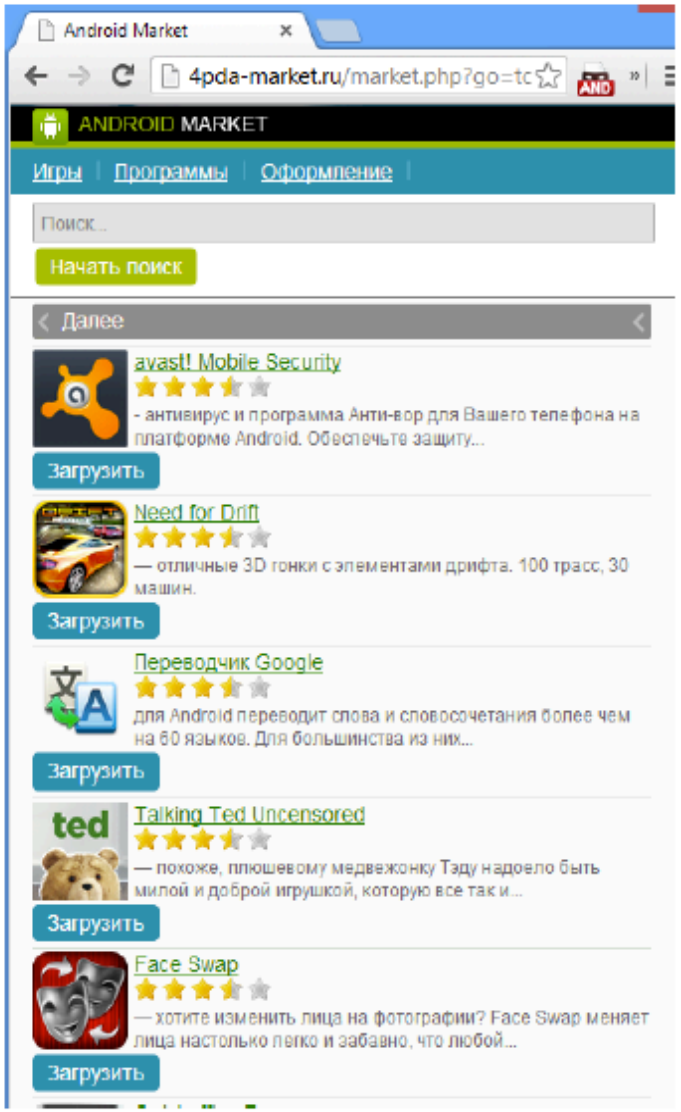
Sicurezza delle App (3)

Esecuzione sicura delle App (Runtime protection):

- Le risorse del sistema e il kernel sono separati (e protetti!) dalle applicazioni utente
- Basato su sistema operativo Linux multi-utente
- Ciascuna applicazione va in esecuzione come un utente diverso
- **Application sandbox**: Ciascuna applicazione in genere viene eseguita con il suo UID nella sua Dalvik/ART virtual machine
  - Protezione di CPU e memoria
  - Comunicazione protetta tramite comunicazione autenticata mediante Unix socket
  - Solo `ping`, `zygote` (spawn another process) eseguiti come `root`

App: tipico attacco

1. Scaricare un'applicazione legittima già presente nel mercato e particolarmente apprezzata
2. Fare il *reverse engineering* del codice
3. Inserire il codice malevolo
4. Ripubblicare in un mercato alternativo l'applicazione con un nome molto simile a quello originale



Screenshot relativo ad un Android Market fasullo

Sicurezza delle App (4)

Permessi delle App:

- La lista dei permessi è contenuta in un file XML, denominato `AndroidManifest.xml`

Fino alla versione 6.0:

- In fase di installazione venivano chiesti i permessi (in gruppo)
  - Ok -> installa
  - No -> non installa

Dopo:

- L'utente ha la possibilità di disattivare alcuni dei permessi concessi alla app in fase di installazione

Sicurezza delle App (5)

Install-time requests (Android 5.1.1 and below)

If the device is running Android 5.1.1 (API level 22) or lower, *or* the app's `targetSdkVersion` is 22 or lower while running on any version of Android, the system automatically asks the user to grant all dangerous permissions for your app at install-time (see figure 2).

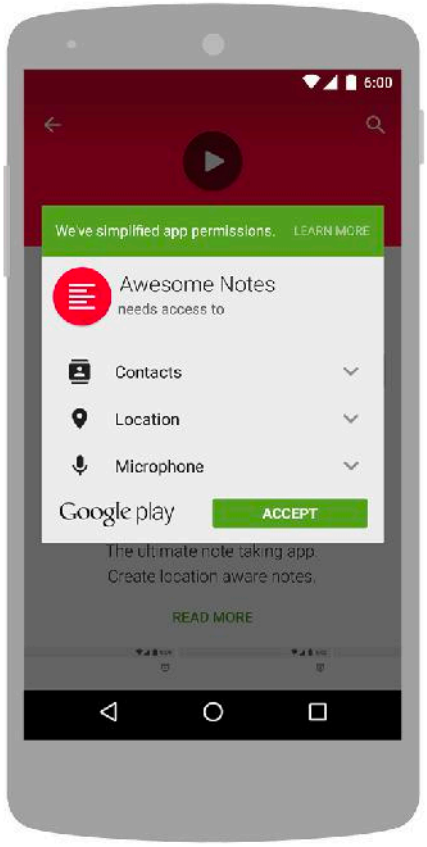


Figure 2. Install-time permission dialog

If the user clicks **Accept**, all permissions the app requests are granted. If the user denies the permissions request, the system cancels the installation of the app.

Sicurezza delle App (6)

Runtime requests (Android 6.0 and higher)

If the device is running Android 6.0 (API level 23) or higher, *and* the app's `targetSdkVersion` is 23 or higher, the user isn't notified of any app permissions at install time. Your app must ask the user to grant the dangerous permissions at runtime. When your app requests permission, the user sees a system dialog (as shown in figure 1, left) telling the user which permission group your app is trying to access. The dialog includes a **Deny** and **Allow** button.

If the user denies the permission request, the next time your app requests the permission, the dialog contains a checkbox that, when checked, indicates the user doesn't want to be prompted for the permission again (see figure 2, right).

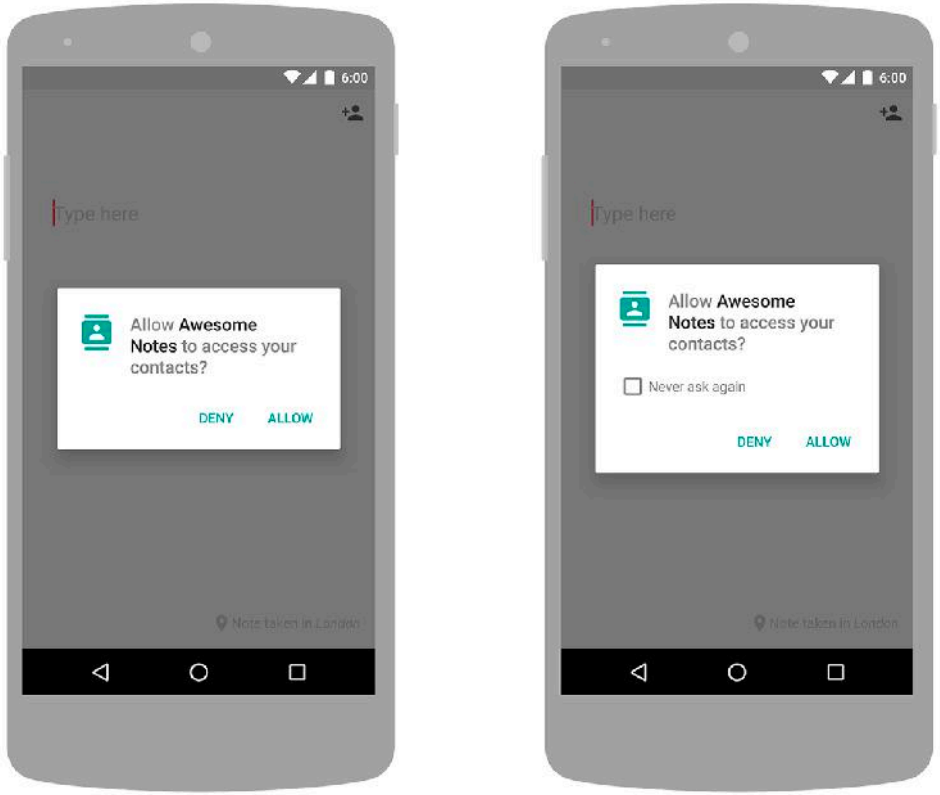


Figure 1. Initial permission dialog (left) and secondary permission request with option to turn off further requests (right)

Sicurezza delle App (7)

Esempio di permessi (dei quasi 200 possibili):

- “android.permission.INTERNET”
- “android.permission.READ_EXTERNAL_STORAGE
- “android.permission.SEND_SMS”
- “android.permission.BLUETOOTH”

È possibile definirne di nuovi, personalizzati.

Sicurezza delle App (8)

App overprivileged

- Spesso le App chiedono <u>più privilegi</u> di quanto necessitano

Brightest Torcia Gratis
GoldenShores Technologies, LLC

Posizione
- posizione approssimativa (basata sulla rete)
- posizione precisa (GPS e basata sulla rete)

Foto/elementi multimediali/file
- modifica/eliminazione di contenuti dell'archivio USB
- lettura dei contenuti dell'archivio USB

Fotocamera
- acquisizione di foto e video

Informazioni sulla connessione Wi-Fi
- visualizzazione connessioni Wi-Fi

ID dispositivo e informazioni sulle chiamate
- lettura stato e identità telefono

Altre
- disattivazione o modifica della barra di stato
- lettura di impostazioni e scorciatoie in Home
- controllo flash
- disattivazione stand-by del dispositivo
- visualizzazione connessioni di rete
- accesso di rete completo
- aggiunta di scorciatoie
- eliminazione di scorciatoie

Data Security (1)

Encryption

Encryption is the process of encoding all user data on an Android device using symmetric encryption keys. Once a device is encrypted, all user-created data is automatically encrypted before committing it to disk and all reads automatically decrypt data before returning it to the calling process. Encryption ensures that even if an unauthorized party tries to access the data, they won't be able to read it.

Android has two methods for device encryption: full-disk encryption and file-based encryption.

Full-disk encryption

Android 5.0 and above supports full-disk encryption. Full-disk encryption uses a single key—protected with the user's device password—to protect the whole of a device's userdata partition. Upon boot, the user must provide their credentials before any part of the disk is accessible.

While this is great for security, it means that most of the core functionality of the phone in not immediately available when users reboot their device. Because access to their data is protected behind their single user credential, features like alarms could not operate, accessibility services were unavailable, and phones could not receive calls.

File-based encryption

Android 7.0 and above supports file-based encryption. File-based encryption allows different files to be encrypted with different keys that can be unlocked independently. Devices that support file-based encryption can also support a new feature called Direct Boot that allows encrypted devices to boot straight to the lock screen, thus enabling quick access to important device features like accessibility services and alarms.

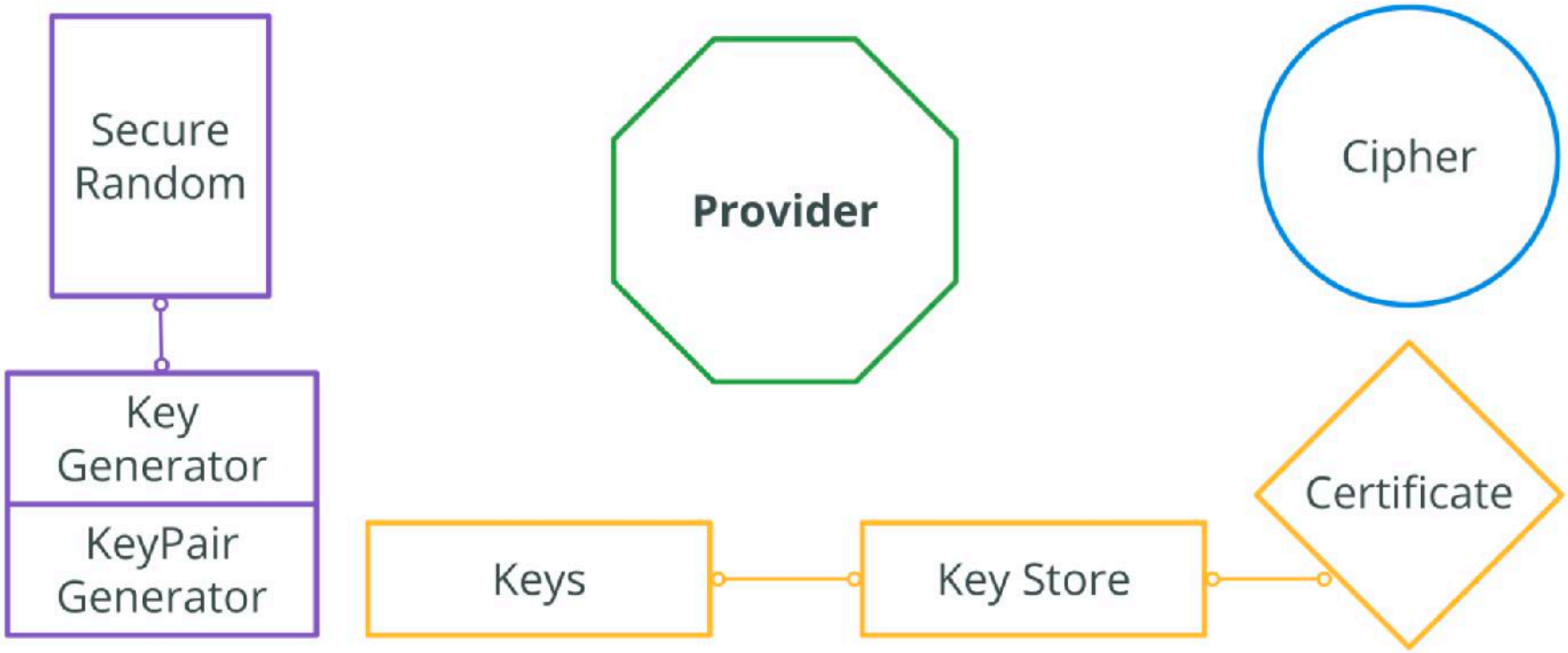
With the introduction of file-based encryption and new APIs to make applications aware of encryption, it is possible for these apps to operate within a limited context. This can happen before users have provided their credentials while still protecting private user information.

https://source.android.com/security/overview/app-security

Data Security (2)

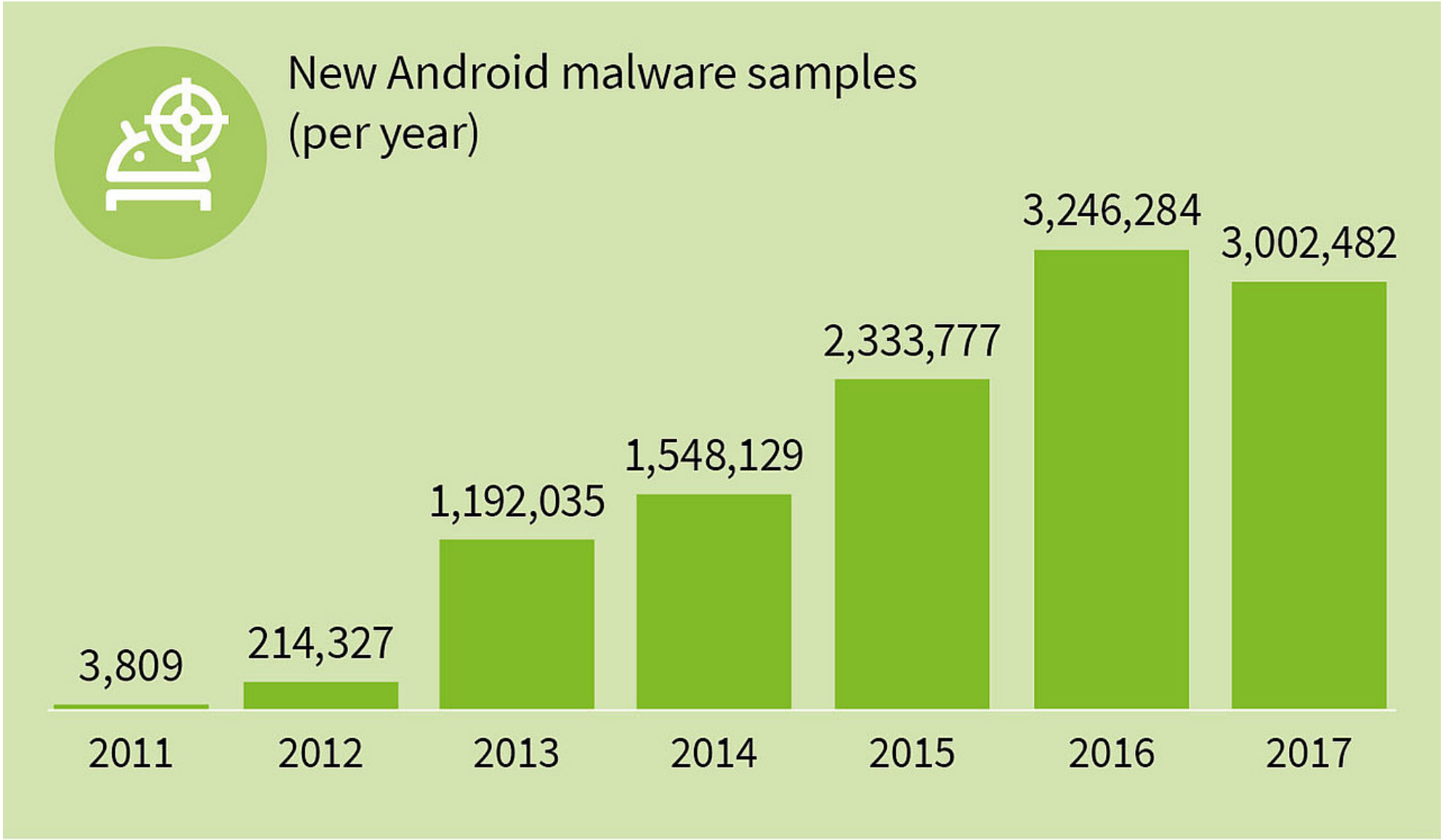
Java Cryptography Architecture

Android builds on the Java Cryptography Architecture (JCA), that provides API for digital signatures, certificates, encryption, keys generation and management.



https://source.android.com/security/overview/app-security

Malware per Android



https://blog.gdatasoftware.com/2017/04/29712-8-400-new-android-malware-samples-every-day

Confronto: Android vs iOS (1)

Sistema operativo

- Unix-based

Linguaggio di programmazione per le App

- Managed execution: Java, .Net
  - quindi no buffer overflow, ecc.
- Native execution: Objective C

Sistema di approvazione e pubblicazione delle applicazioni sul mercato

- Mercato: Aperto vs Controllato dal venditore
- Firma delle App: Self-signed vs Vendor-issued

Confronto: Android vs iOS (2)

Application permission

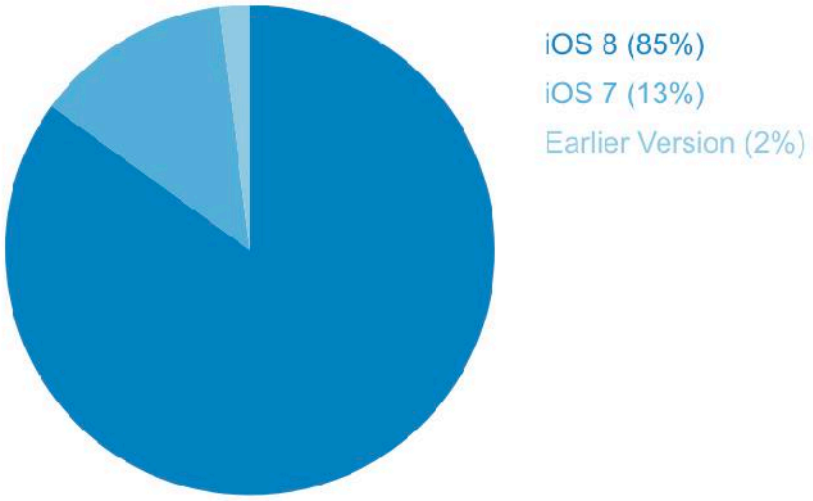
- I permessi di Android si basano sul Manifesto
- Tutte le app iOS hanno lo stesso insieme di privilegi "sandbox"

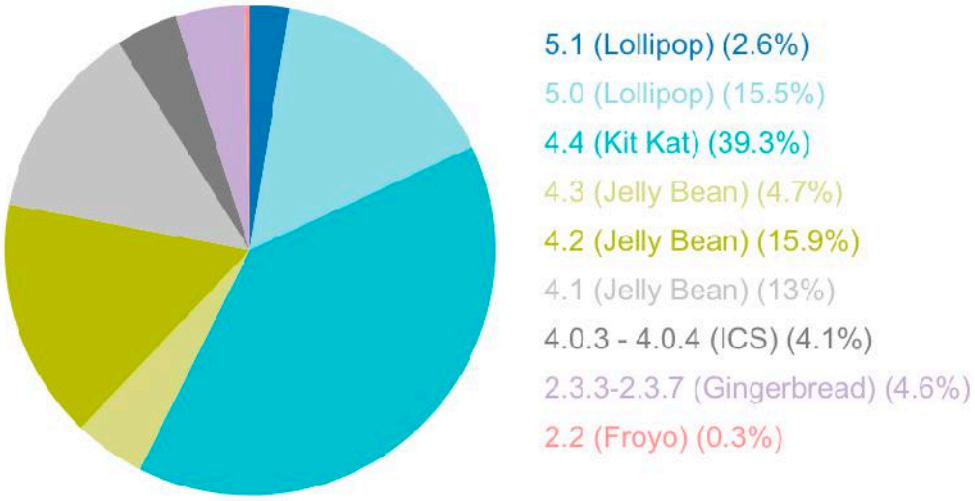
Harware vendors

- Multiple vs one

Confronto: Android vs iOS (3)

Frammentazione?





PROBLEMI DI SICUREZZA CROSS-PLATFORM

Ma davvero i nostri dati sono sicuri?

Problemi:

- Siamo consapevoli di quali *condizioni* accettiamo quando installiamo una App?
- Sappiamo veramente *quali* (e *quanti*) dati vengono raccolti dalle App?
- Siamo sicuri che i nostri dati vengano gestiti (i.e., anche *trasmessi*) *in modo sicuro*?
- Siamo sicuri che alcuni dei nostri dati siano proprio *anonimi*?

Caso di studio 1: mHealth App

I dati sanitari sono dati sicuramente sensibili

Avete mai pensato quando usate Nike+ Training Club o BeFit:

- Dove e come vengono salvati i vostri dati?
- Come vengono trasmessi i dati dal sensore al cellulare e dal cellulare al cloud?
- Chi e come gestisce i **vostri** dati?

Alcuni studi hanno mostrato che…

- 2014, studio del *US Federal Trade Commission* (FTC): analisi di 12 App, spedivano a 76 terze parti screen size, modello del dispositivo, e setting della lingua. Ad altri Unique Device Identifier (UDID), MAC address e IMEI, o informazioni personali degli utenti (i.e., percorsi di running, abitudini alimentari, pattern del sonno, ogni quanto si muovono, sesso, localizzazione e codice postale)
- 2013, studio del procuratore distrettuale della California, analizzate 43 App: "**13 percent of them encrypted all data** between the app and the developer's website, a third of the apps were found transmitting user information to a party not disclosed by the developer or the developer's website"

Caso di studio 2: Studio del WSJ

Analizzate 20 App:

- 17 App spediscono a 70 terze parti informazioni precise di locazione
- Tra le terze parti hanno identificato alcune aziende che vendono i dati di locazione in loro possesso
- Partendo da i dati (anonimi) recuperati dalle applicazioni sono riusciti a risalire all'identità di alcune persone

Caso di studio 3: Importanza dei Metadati (1)

Studio del 2014 (già visto in precedenza): Quali informazioni rivelano i dati di chiamate telefoniche?

- 823 partecipanti (maggiorenni, con cellulare Android e account su Facebook)
- Dati di chiamate e sms recuperati da un App per Android usando file di log e API standard)
  - 251,788 chiamate e 1,234,231 sms
- Dati raccolti da cellulare:
  - Data e ora
  - Chiamata/sms in entrata o in uscita e numero dell'altro telefono
  - Durata in secondi o lunghezza in caratteri
- Dati raccolti da profilo Facebook:
  - Informazioni personali contenute nell'account Facebook (genere, status, occupazione, città di residenza, interessi, fede politica e religiosa)

Caso di studio 3: Importanza dei Metadati (2)

Informazioni dedotte:

- Reidentificazione *automatica* e *manuale* dei numeri
- Inferenza sulla relazione amorosa

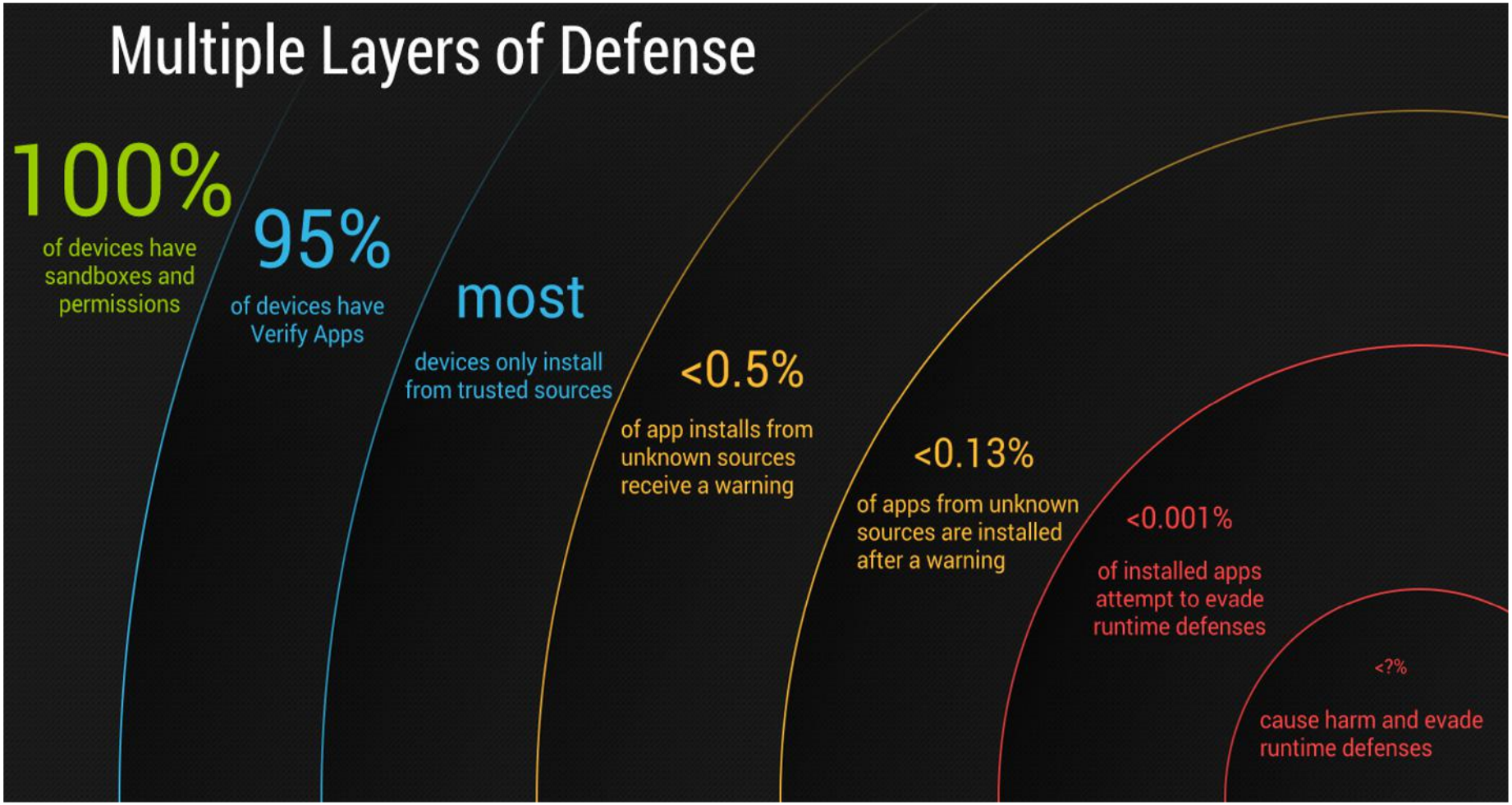
Table 1. Performance of telephone number reidentification (automated approaches)

Look-up source	Matched, %
Google Places	16.6
Yelp	10.5
Facebook	13.7
All Automated Sources	31.9

Table 2. Performance of telephone number reidentification (manual and combined approaches)

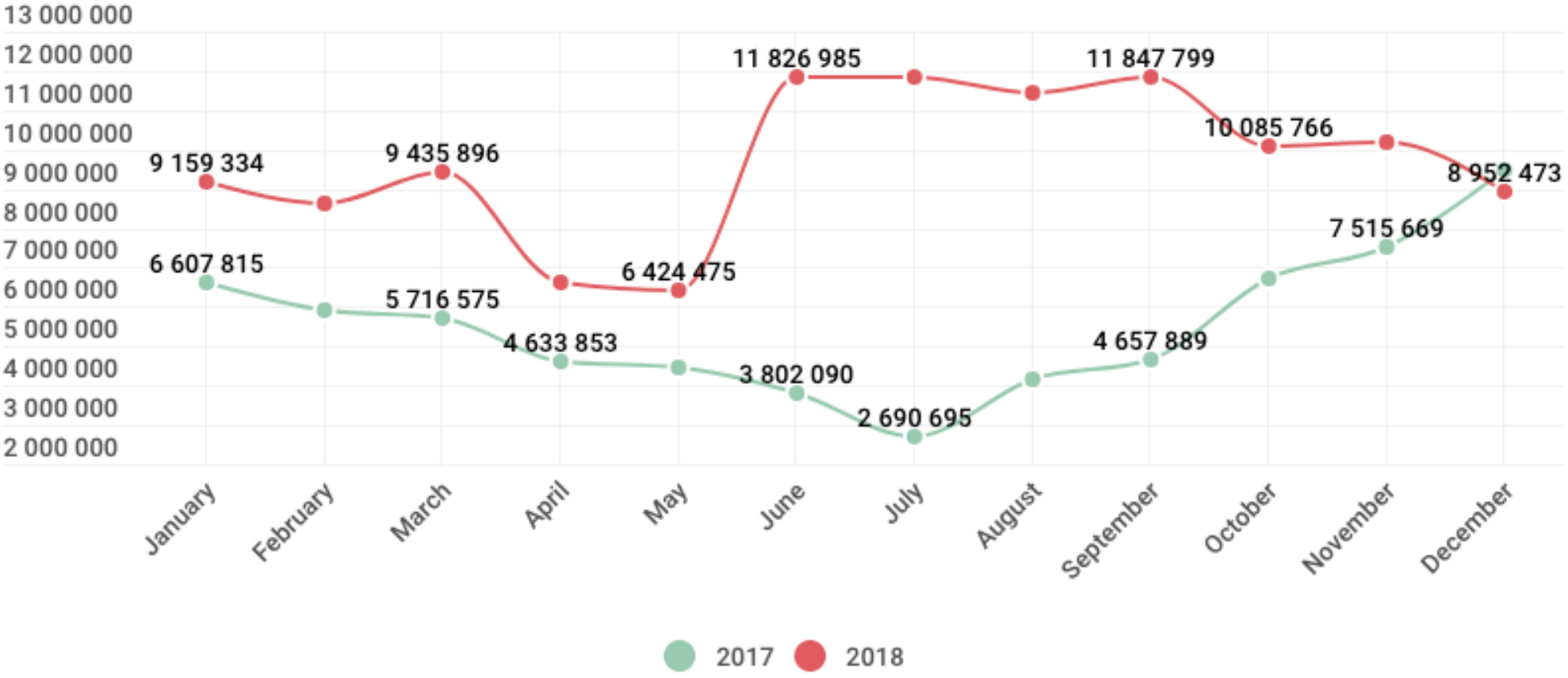
Look-up source	Matched, %
Intelius	65
Google search	58
All automated sources	26
All sources	82

Esistono mobile malware?

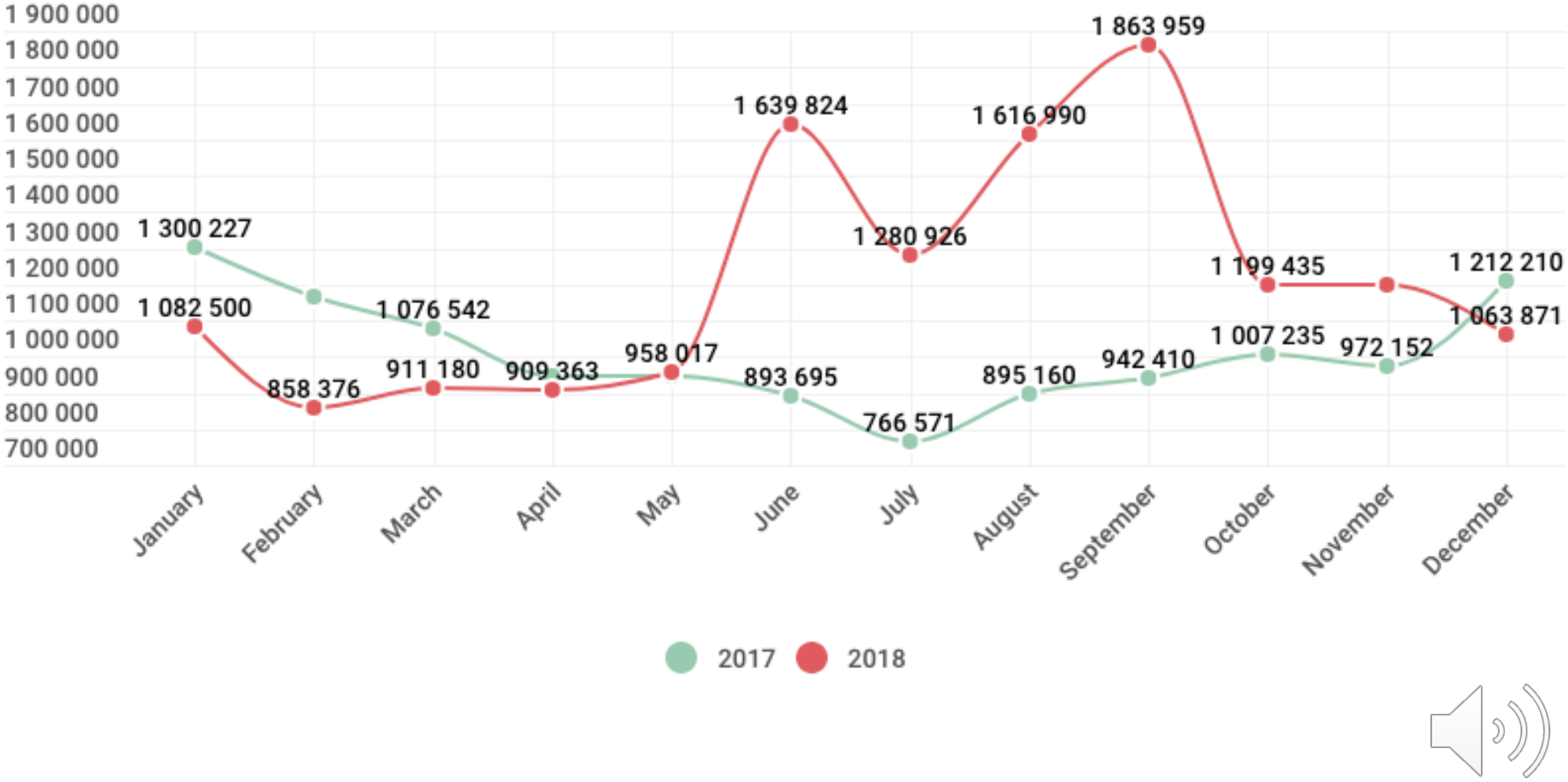


Fonte: Google su Andr d

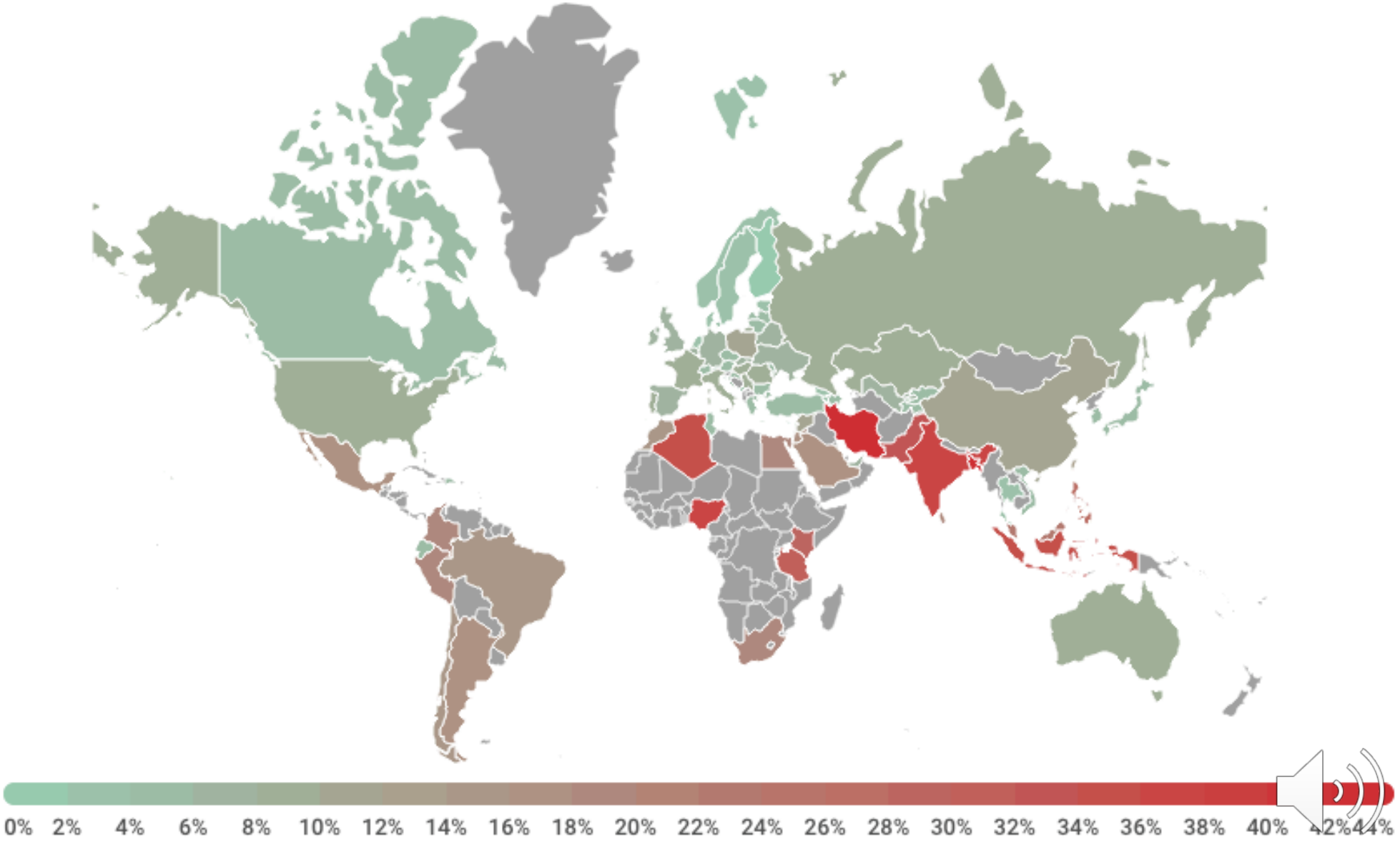
Mobile malware (1) – Attacchi rilevati



Mobile malware (2) – Numero di vittime



Mobile malware (3) – Distribuzione vittime



PROBLEMI DI SICUREZZA CROSS-PLATFORM: *MALWARE PER DISPOSITIVI MOBILI*

Cosa è un *malware*?

Malware = Malitious + Software

- Software con intento malizioso
- Sequenza di codice progettata per **danneggiare intenzionalmente** un sistema, i dati che contiene o comunque alterare il suo normale funzionamento, all'insaputa dell'utente
  - In genere deve sfruttare una vulnerabilità del sistema per poter venire installato/diffondersi

Obiettivi del malware

- Monetizzare
  - Furto (e vendita) di informazioni (dati personali, credenziali)
  - Richiesta di soldi (Ransomware)
  - Invio di Premium SMS
  - Bitcoin mining
  - Advertising
- Cancellare file
- Rubare i numeri seriali del software
- Usare il computer attaccato come *relay* (ripetitore) per altri attacchi
- Targeted Attack: attacchi a un insieme specifico di individui (spionaggio industriale, ecc.)

Tipologie di malware

Virus

Worm

Trojan horse

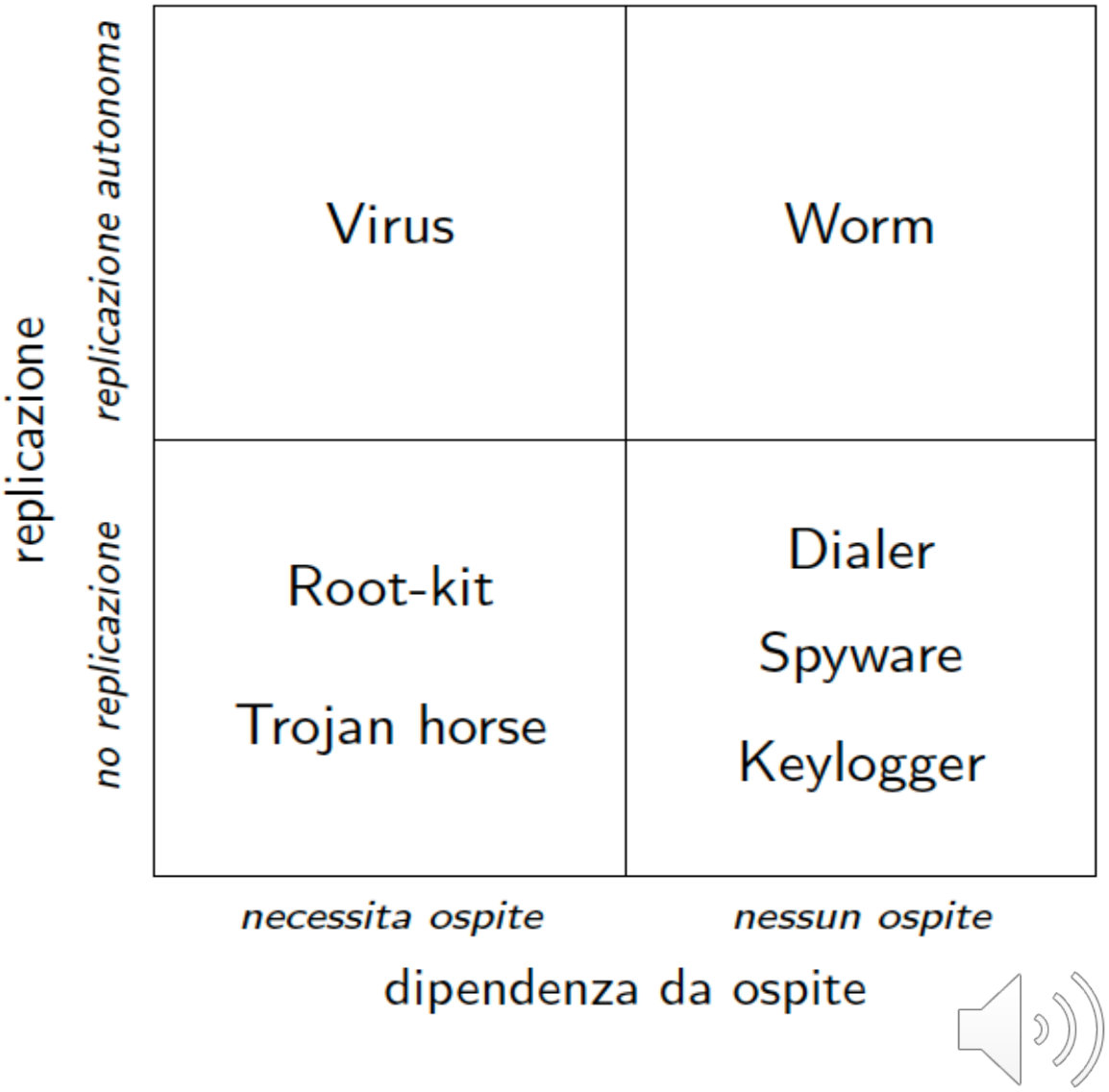
Spyware

Adware

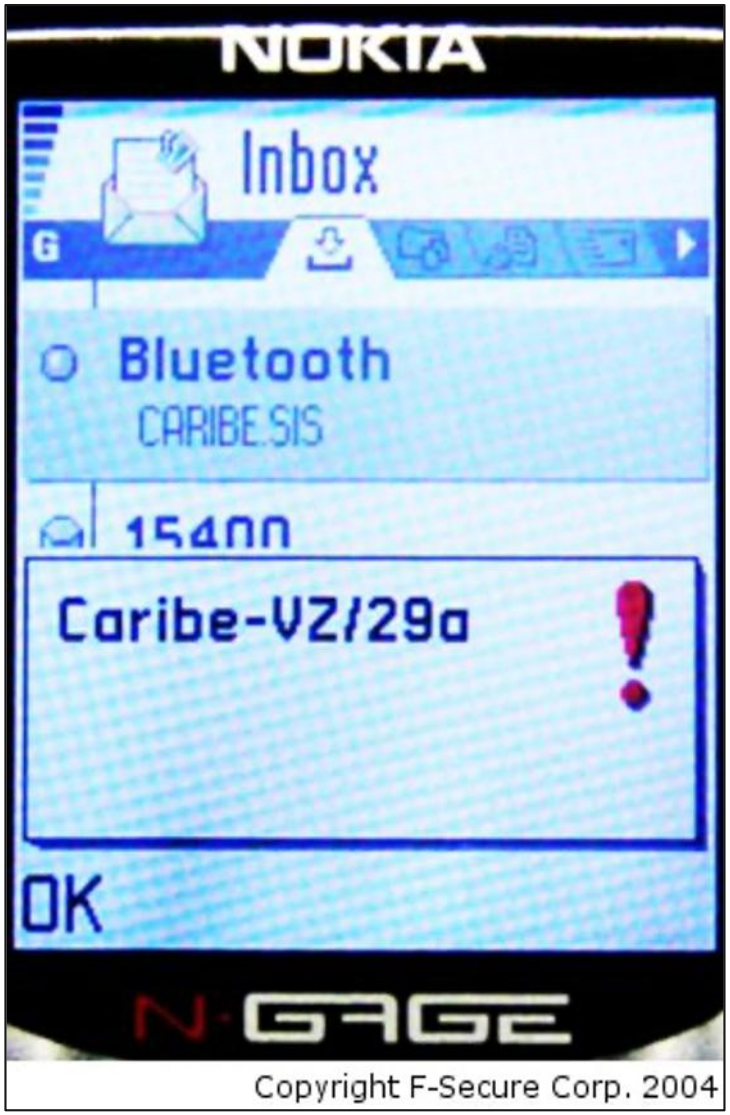
Rootkit

Keylogger

Botnet



Cabir (2004)

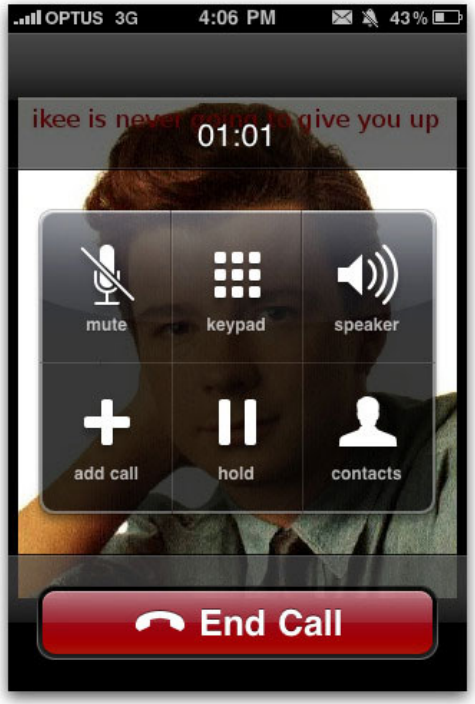


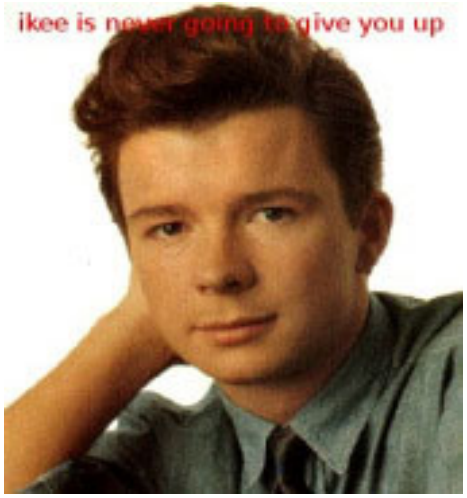
- Il primo malware per mobile in assoluto
- Destinato a dispositivi con Symbian OS
- Si diffondeva come SIS file (Symbian OS distribution file) e visualizza il messaggio "Caribe"
- Esauriva la batteria del telefono in modo estremamente rapido
- Cercava di diffondersi mediante Bluetooth

Skull (2004)

- Destinato a dispositivi con Symbian OS
- Corrompeva i file legati a funzionalità critiche
  - SMS / MMS
  - Web browsing
  - Macchina fotografica
- Sostituiva tutte le icone con dei teschi

Ikee (2009)





- Destinato a dispositivi iOS
- Diffusosi solo in Australia
- Funziona solo su cellulari jailbroken con installato ssh
- Cambia l'immagine di background con l'immagine di Rick Astley, un cantante pop degli anni '80

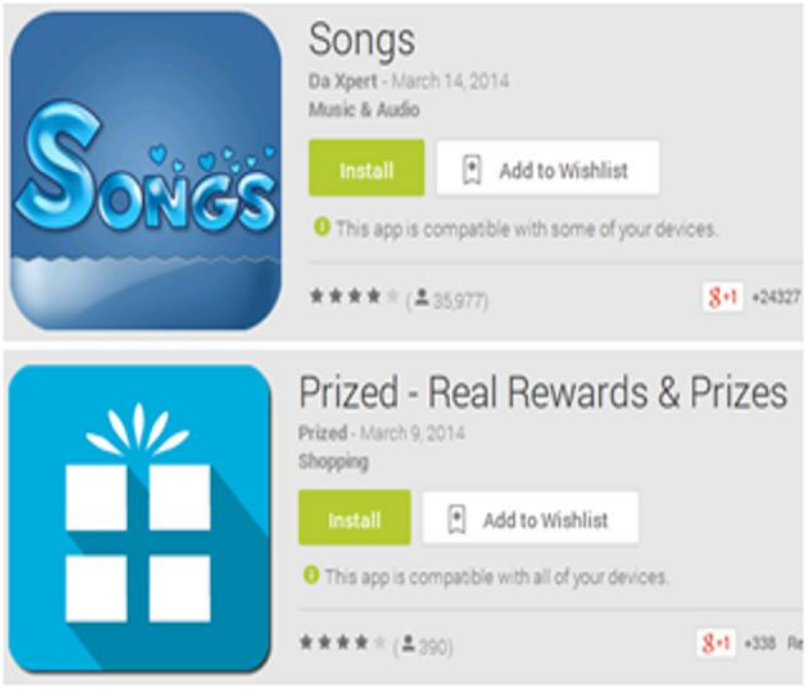
HippoSMS (2011)

- Spedisce SMS dal cellulare verso numeri a pagamento
- Non tracciabile: tutti gli SMS legati al malware vengono poi rimossi

DroidDream (2011)

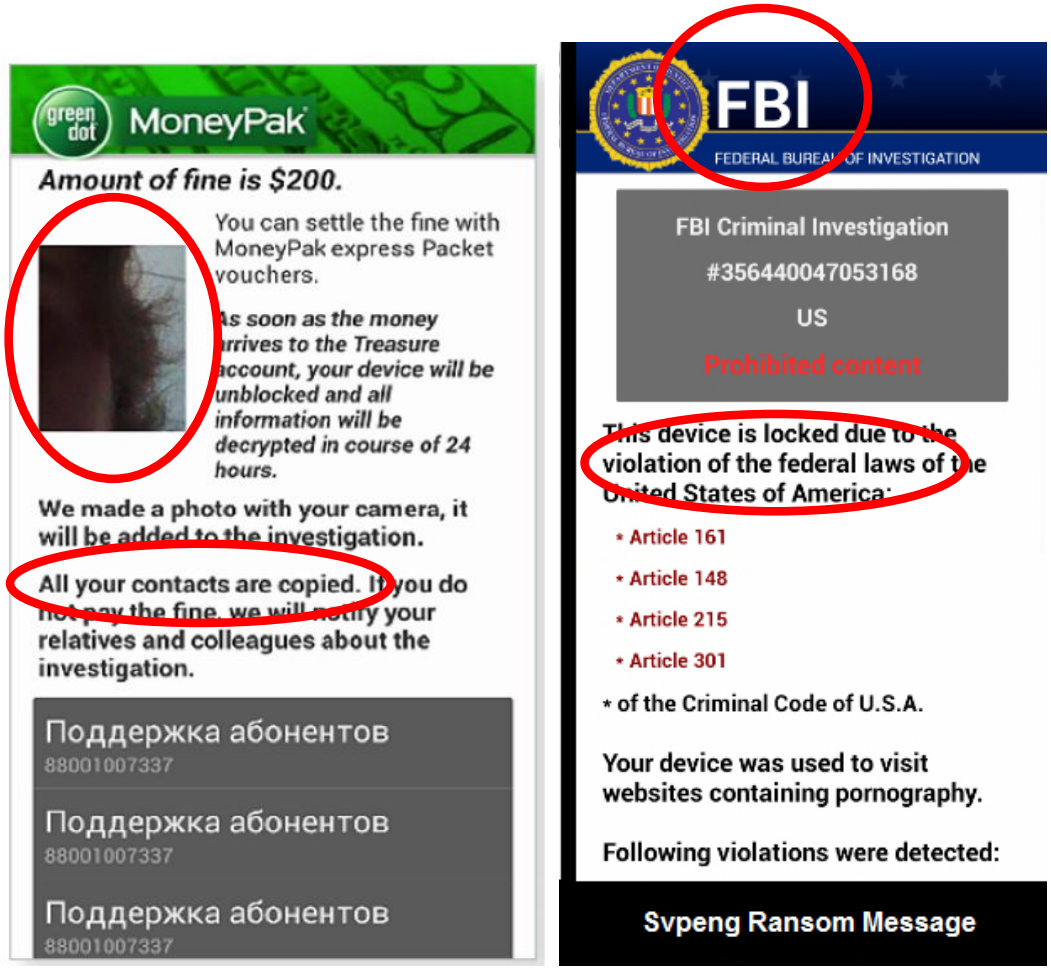
- Destinato a dispositivi Android
- **Contenuto in più di 58 apps caricate sul Google Market**
- Funzionamento (con l'obiettivo di monetizzare):
  - Ruba dati
  - Spedisce credenziali agli attaccanti
  - Scarica altri programmi maliziosi (che possono permettere all'attaccante di avere accesso al dispositivo)

Bitcoin Miner (2014)



- App legittime modificate per fare mining di bitcoin in background
- Per l'attaccante poco lavoro:
  - l'App modificata non deve venire riscritta
  - il codice per fare minig rubato da un'altra App

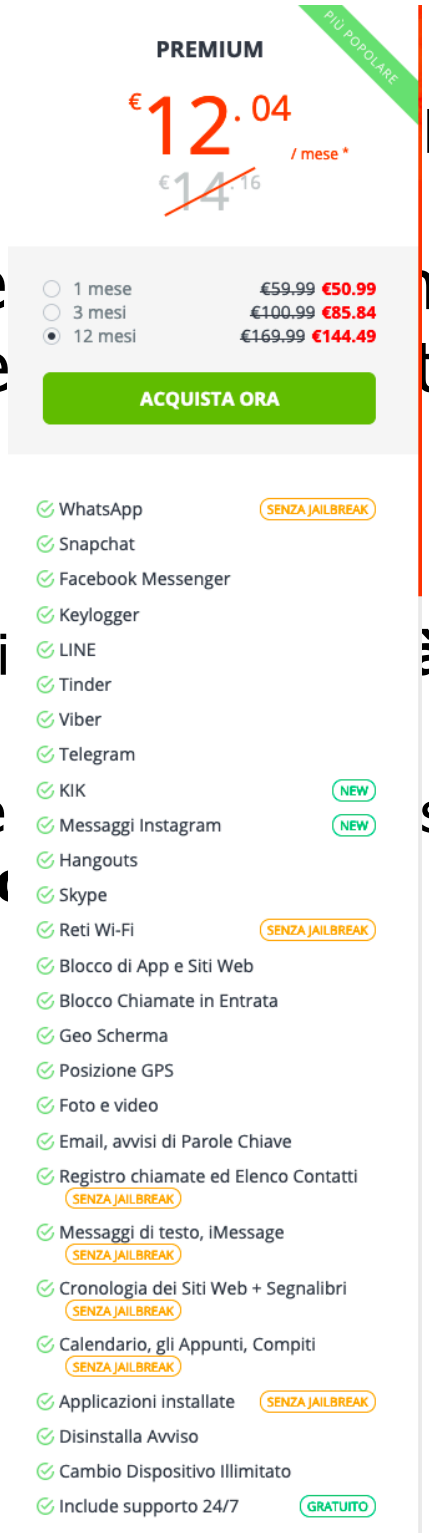
Ransomware



- *Limita l'accesso del dispositivo che infetta*, richiedendo un riscatto da pagare per rimuovere la limitazione
  - Cifra i file con crittografia asimmetrica (chiave privata nota solo all'attaccante)
  - Modifica il master boot record e impedisce l'avvio del SO
- Spaventa e "mette pressione" all'utente
  - FBI
  - Violazione di leggi

re (1)

- Software usato per raccogliere ni dal sistema su cui sono installati e tterle ad un destinatario interessato
- Ma è lecito?
  - la **commercializzazione** è consi à non **vietata dalla legge**
  - il **concreto utilizzo** del software sanzione nel caso di **assenza di consens**



Spyware (2)

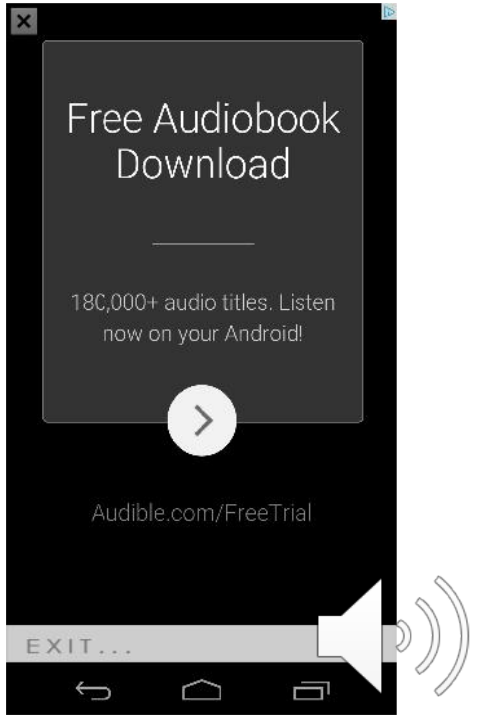
- RAT (Remote Administration Software)
  - Permette di controllare da remoto un sistema, come se si avesse un accesso fisico
  - Necessita di una connessione di rete

- Esempio: RCSAndroid (Remote Control System Android)
  - Sviluppato da Hacking Team e venduto a governi per targeted attack
    - Registrava audio usando il microfono del dispositivo, le password (WiFi e di account online), video, ecc..
  - Ironia della sorte: anche HT ha subito un attacco e tutti i dati rubati (mail) sono finiti su WikiLeaks

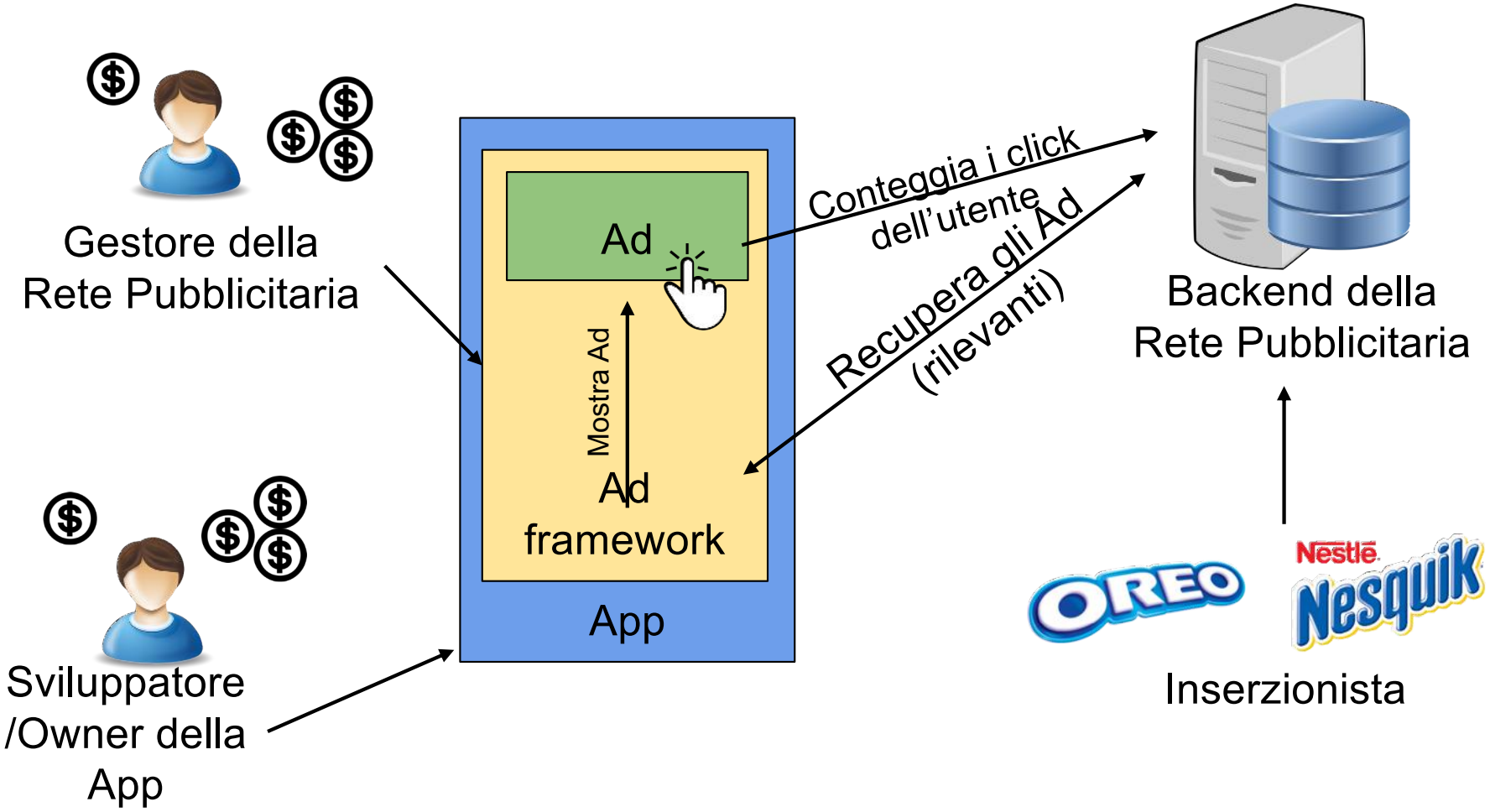
Adware (o Annoyware)

- Adware o advertising-supported software: software che presenta all'utente messaggi pubblicitari durante l'uso
  - causa rallentamenti del PC (e spesso comunica le abitudini di navigazione ad un server remoto)

- Tecnicamente non è una frode, però esistono malware che utilizzano la business logic che sta alla base degli di adware a scopo di lucro...



L'Ecosistema della pubblicità su Mobile



Ad = Advertisement = Messaggio pubblicitario

Attacchi all'Ad Ecosystem

- La App simula i click dell'utente
  - Possibile visto che App e Ad Framework sono nello stesso sandbox!
- La App usa più Ad Framework sovrapposti, tanto l'inserzionista e il gestore della Rete Pubblicitaria non se ne accorgono
- *Click farm:* Gruppi di persone sottopagate per fare click su inserzioni pubblicitarie!!!

Malware – Come infettare un dispositivo?

- Bug del Sistema operativo
- Browser Web
- Link maliziosi su siti web e/o social network (Phishing)
- Smartphone-based:
  - SMS/MMS
    - Trojan SMS
  - Canale di comunicazione: WiFi, Blutooth, GSM
  - **App Infette**

App Infette?

Bisogna bypassare diversi meccanismi di sicurezza!

1. Controllo (manuale e automatico) prima che venga inserita nello store
   - Code obfuscation
   - Dynamic code loading
   - Pubblicazione su uno store di terze parti
2. Ogni volta che la App necessita dei permessi particolari deve chiedere conferma all'utente
   - Il malware può bypassare i controlli dei permessi o fare privilege escalation
3. L'utente deve scegliere di installare una App
   - Social engineering
   - Repackaging
   - Trojan App

Social Engineering per infettare App

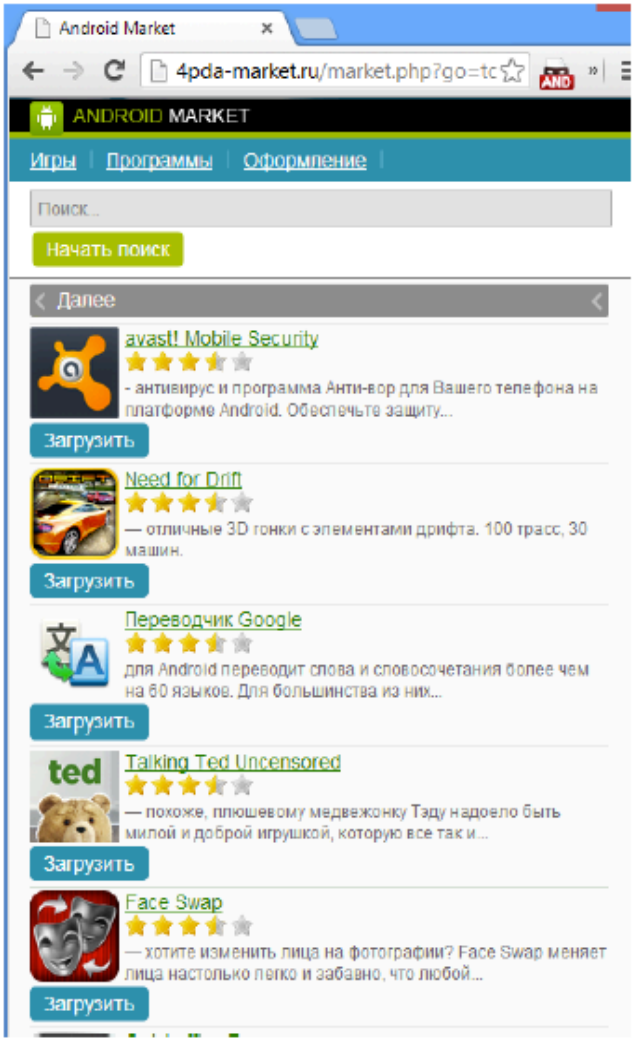
Come convincere un utente a scaricare una App?

- Inserire nello store una App simile (nome o icona)
- Mettere delle inserzioni maliziose che puntino alla App sbagliata (quella con il malware, non l'originale)
- Offrire versioni gratis di App a pagamento
- Offrire funzionalità aggiuntive rispetto a quelle offerte alla App di base

App Repackaging

1. Scaricare un'applicazione legittima già presente nel mercato e apprezzata
2. Fare il *reverse engineering* del codice
3. Inserire il codice malevolo
4. Ripubblicare in uno store alternativo l'applicazione con un nome molto simile a quello originale

Tecnicamente semplice!

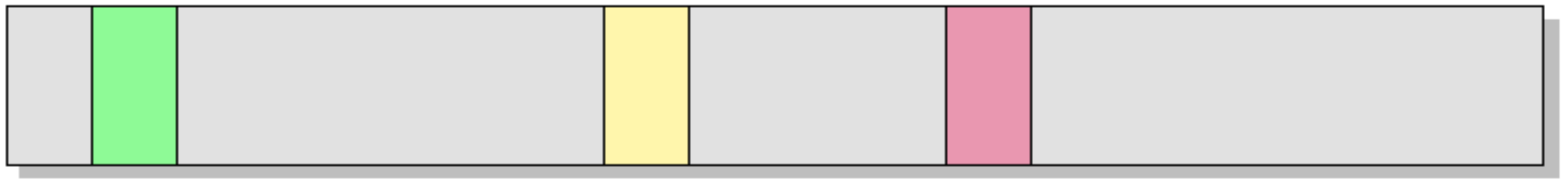


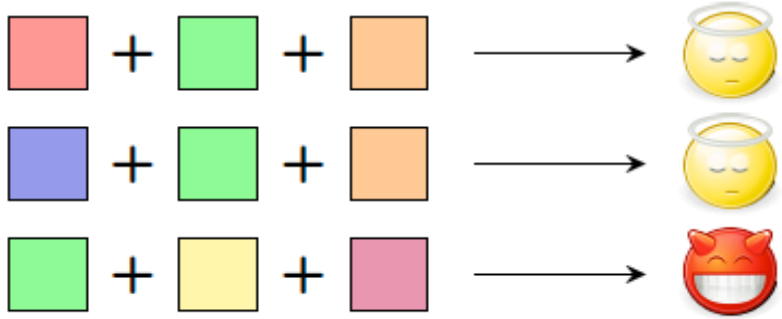
Screenshot relativo ad un Android Mark[illegible]suNo

Come difendersi?

Antivirus:

- Database di *signature* (una stringa che possa identificare un virus)
- Un'applicazione che contiene una *signature* nota viene considerata infetta





Grazie.

www.vincenzocalabro.it
LinkedIn vincenzocalabro