Sicurezza dei sistemi mobili

Sicurezza degli smartphones

Attacchi negli smartphones

Rispetto ai computer gli elementi piu' vulnerabili ad un attacco in uno smartphone sono:

- Browser Web;
- App (Google Play Store, App Store);
- Sistema operativo;
- SMS/MMS;
- Canali di comunicazione: WiFi, Bluetooth, GPS.

Obiettivi

- Dati confidenziali:
  - Numeri di carta di credito;
  - Credenziali per l'autenticazione;
  - Informazioni private (foto, files, ecc);
  - Log delle attivita' (calendario, chiamate, ecc).
- Identita': rubare l'identita' del proprietario del cellulare;
- Disponibilita': limitare l'accesso al cellulare.

In particolare e' possibile effettuare alcuni attacchi specifici:

- Identificazione della posizione;
- Registrazione delle chiamate telefoniche e SMS;
- Chiamate/SMS a numeri a tariffazione maggiorata di proprieta' dell'attaccante ma registrati anonimamente.

Oppure attacchi simili ai pc:

- Rubare dati;
- Malvertising (malicious advertising);
- Phishing.

Malware

Un malware deriva dalla crasi dei termini "malicious" e "software" quindi si tratta di una sequenza di codice progettata per danneggiare intenzionalmente un sistema, i dati che contiene, o comunque alterare il suo normale funzionamento all'insaputa dell'utente. La diffusione di questi software e' in continuo aumento.

Tipologie di malware

- No replicazione:
  - Necessita di ospite:
    - Root-kit;
    - Trojan Horse;
  - Non necessita di ospite:
    - Dialer;
    - Spyware;

        - Keylogger;
- Replicazione autonoma:
    - Necessita di ospite:
        - Virus;
    - Non necessita di ospite:
        - Worm.

Virus

Il virus e' un software malevolo che infetta altri programmi includendo una versione evoluta, modificata, di se stesso.

Si replica in modo autonomo e necessita di un ospite in quanto viene eseguito solo (e ogni volta) se il file infetto viene aperto.

La sua trasmissione avviene tramite lo spostamento del file infetto.

Esistono virus **polimorfici**, che "mutano", mantenendo comunque sempre uguale la loro "impronta virale" ovverlo la sequenza di byte che identifica in maniera univoca un virus.

Altri virus sono detti **metamorfici**, ossia che cambiano ad ogni infezione.

Trojan Horse

Sono una classe di software che hanno funzionalita' maligne, ma camuffate coem software con funzionalita' lecite e talvolta utili (il che induce gli utenti ad usarli).

Necessitano di un ospite, quindi di un file che deve essere aperto, ma non si replicano in modo autonomo. Per la diffusione e' dunque necessario inviarli consapevolmente alla vittima, oppure tramite applicazioni con inserimento di funzionalita' da terze parti (**plugins**).

Worm

E' un programma che non ha bisogno di inferttare altri file per diffondersi perche **modifica il sistema operativo** della macchina ospite in modo da venire eseguito automaticamente.

In genere tenta di replicarsi tramite Internet, spedendo copie di se stesso ad altri hosts.

Scareware

Con questo termine vengono identificate diverse classi di **scam software** spesso con benefici limitati, venduto ai consumatori per motivi di marketing.

- **Spyware**: software usato per raccogliere informazioni del sistema su cui sono installati per trasmetterle ad un destinatario interessato (cronologia navigazione, password, chiavi crittografate di un utente, ecc);
- **Adware**: software che presenta all'utente messaggi pubblicitari durante l'uso a fronte di un prezzo ridotto o nullo. Causa rallentamenti del PC e spesso comunica informazioni del computer dell'utente ad un server remoto come lo spyware.

Infezione dei file

In genere i malware hanno come target un file, sostituendo una porzione di codice tramite:

- Sovrascrittura;
- Inserimento in testa;
- Inserimento in coda:
- Inserimento in una "cavity";
- Inserimento in una "multi-cavity".

Esempi

- Su Windows:
    - Cartella auto-start;

    - Win.ini: settings usati in fase di boot;
    - System.ini: carica i driver dei vari dispositivi e delle shell;
    - Config.sys: configura i componenti HW del pc.
- Su Unix/Linux:
    - Init.d: contiene una serie di script per far avviare/terminare vari servizi del sistema;
    - /etc/rc.local: in esecuzione alla fine di tutti i livelli di boot multi-user;
    - Crontab: contiene istruzioni per il daemon cron.

Propagazione di un file infetto

- Cartelle condivise;
- Allegato o link in una mail;
- Dirottamento del browser: inserimento del link al file infetto nei risultati di un motore di ricerca;
- P2P.

Come difendersi

- Antivirus:
    - Analizzano il comportamento del sistema;
    - Analizzano i binari per decidere se si tratta di un virus;
    - Gli antivirus hanno un **database di signature** che permette all'antivirus di riconoscere (se presenti) la fingerprint di un virus in un file/applicazione.
    - Utilizzo di **euristiche**: analizzando il comportamento:
        - Accessi alla rete;
        - Apertura dei file;
        - Tentativi di cancellazione dei file;
        - Tentativi di modifica al boot-sector.
    - Utilizzo di **checkstum**:
        - Calcolo dei checksum per riconoscere binari originali e per i file di configurazione.
- Sandbox Analysis: eseguire gli eseguibili in una VM per studiarne il comportamento.

Underground Economy

Esiste un mercato nero per la rivendita di informazioni personali obiettivo dei malware.

Sicurezza mobile

Piattaforma iOS

Ha un'architettura a livelli implementata in C/Objective-C (Swift):

- **Cocoa Touch**: contiene il **Foundation framework**, un'insieme di classi per Objective-C, e l'**UIKit** che offre le funzionalita' principali per costruire le app;
- **Media layer**: supporto 2D e 3D drawing, audio e video;
- **Core Services e Core OS**: API per file, rete, include SQLite, thread POSIX, socket UNIX;
- **Kernel**: versione light del kernel di Mac OS.

Le app vengono realizzate in Objective-C (Swift) usando l'Apple SDK. Il modello di gestione degli eventi e' basato su **touch event** e i servizi di base usati da tutte le applicazioni sono definiti nel Foundation framework e l'UIKit.

Sicurezza su iOS

Per garantire la sicurezza delle App, a runtime le risorse del sistema e il kernel sono separati dalle applicazioni utente.

Le app vengono eseguite in una sorta di **sandbox** che nega l'accesso ai dati di altre app. Ciascuna app ha una cartella sandbox per limitare le conseguenze delle app, inoltre garantisce a tutte le app gli stessi privilegi.

La comunicazione tra le app avviene solo mediante iOS API definite dal sistema.

Le app devono venire firmate da un certificato emesso da Apple, quindi si tratta di un **certificato Vendor-Issued**.

Piattaforma Android

- **Application layer**: fornisce le funzionalita' aggiunte dalle applicazioni;
- **Android Runtime**: contiene le **Core libraries** scritte in Java e la **Dalvik VM** che esegue i file in un formato piu' efficiente e compatto rispetto ai file .class.
- **Libraries**: contengono le librerie native di Android che interagisce con l'Android Runtime;
- **Linux Kernel**: gestisce servizi di sistema fondamentali come accesso ai file, rete, memoria, processi, ecc;

Android dispone di software per la comunicazione sicura sulla rete e una libreria di primitive crittografiche scritte in Java.

Sicurezza su Android

Le applicazioni vengono scritte in Java e pacchettizzate in un archivio .apk **Self-Signed**. I permessi delle app sono stabiliti in fase di installazione.

Il mercato e' quindi aperto anche ad applicazioni maliziose.

Su Android l'isolamento delle app rispetto al sistema viene ottenuto tramite:

- Esecuzione delle applicazioni tramite **utenti diversi** (uno per ogni applicazione) sfruttando il sistema Linux multi-utente;
- Ciascuna app viene eseguita con il suo User ID nella Dalvik VM:
  - Garantendo la protezione di CPU e memoria;
  - La comunicazione e' protetta tramite comunicazione autenticata mediante socket Unix;
  - Solo ping, zygote (spawn dei processi) sono eseguiti come root.
- La comunicazione tra applicazioni avviene solo se le due applicazioni condividono lo stesso Linux User ID: in tal caso accedono reciprocamente ai file e possono condividere lo stesso processo Linux e Dalvik VM.

Attacco tipico alle app

- Scaricare un'applicazione legittima gia' presente nel mercato e particolarmente apprezzata;
- Effettaure il reverse-engineering del codice;
- Inserire codice malevolo;
- Ripubblicare l'applicazione con un nome molto simile a quello originale.

Confronto iOS e Android

	iOS	Android
Approvazione delle App	Mercato controllato dal venditore di app controllate: Vendor-Issued	Mercato aperto: Self-Signed
Permessi delle App	Tutte le app hanno gli stessi privilegi	Decisi in fase di installazione
Linguaggio di programmazione	Objective-C / Swift	Java (protezione da buffer-overflow)