Mobile Platform Security Models

John Mitchell

Outline
- Introduction
- Apple iOS security model
- Android security model
- Windows 7 Mobile security model

Mobile phone market share

Mobile Operating Systems

Many mobile apps

Source: IBM X-Force, Mar 2011
Two attack vectors

- Web browser
- Installed apps

Both are increasing in prevalence and sophistication

Mobile malware attacks

- Unique to phones:
  - Premium SMS messages
  - Steal location
  - Record phone calls
  - Log SMS
- Similar to desktop/PCs:
  - Connects to botmasters
  - Steal data
  - Phishing
  - Malvertising

Mobile malware

- Dec. 2009: Developer 09Droid uploads ~50 apps that impersonate banks
- Phishing/steal user credentials
- Mar/May 2011: DroidDream, DroidDreamLite re-package legitimate apps
- June 2011: Trojan impersonates Angry Birds plug-in

Mobile malware examples

- DroidDream (Android)
  - Over 58 root'ed apps uploaded to Google app marketplace
  - Conducts data theft; send credentials to attackers
- Ikee (iOS)
  - Experimental; “rick-rolled” the phone
  - Worm capabilities (targeted default ssh pwd)
  - Worked only on jailbroken phones with ssh installed (could have been worse)
- Zitmo (Symbian, BlackBerry, Windows Mobile, & Android)
  - Captures mTANs from SMS messages; aimed at defeating 2-factor auth
  - Works in conjunction with Zeus botnet; sent in conjunction with user PC infection
  - Propagates via SMS; claims to install a “security certificate”

Outline

- Introduction
- Apple iOS security model
- Android security model
- Windows 7 Mobile security model

Differences between platforms

- Operating system
  - Unix
  - Windows
- Approval process for applications
  - Market: Vendor controlled/Open
  - App signing: Vendor-issued/self-signed
- Programming language for applications
  - Managed execution: Java, .Net
  - Native execution: Objective C
Apple iOS

- From: iOS App Programming Guide

iOS Platform

- Kernel: based on Mach kernel like Mac OS X
- Core OS and Core Services: APIs for files, network, ... includes SQLite, POSIX threads, UNIX sockets
- Media layer: supports 2D and 3D drawing, audio, video
- Cocoa Touch: Foundation framework, OO support for collections, file management, network operations; UI Kits

Implemented in C and Objective-C

iOS Application Development

- Apps developed in Objective-C using Apple SDK
- Event-handling model based on touch events.
- Foundation and UIKit frameworks provide the key services used by all iOS applications

Apple iOS Security

- Device security: Methods that prevent unauthorized use of the device
- Data security: Protecting data at rest, even when a device is lost or stolen
- Network security: Networking protocols and the encryption of data in transmission
- App security: Secure platform foundation


Device Security

- Strong passcodes
- Passcode expiration
- Passcode reuse history
- Maximum failed attempts
- Over-the-air passcode enforcement
- Progressive passcode timeout

Data Security

- Hardware encryption
- Data protection
- Remote wipe
- Local wipe
- Encrypted Configuration Profiles
- Encrypted iTunes backups
Network Security

- Built-in Cisco IPSec, L2TP, PPTP VPN
- SSL VPN via App Store apps
- SSL/TLS with X.509 certificates
- WPA/WPA2 Enterprise with 802.1X
- Certificate-based authentication
- RSA SecurID, CRYPTOCard

Security Services Framework

- Generic Security Services framework (GSS.framework) provides a standard set of security-related services to iOS applications.
  - IETF RFC 2743 and RFC 4401
    - http://www.ietf.org/rfc/rfc2743.txt

App Security

- Runtime protection
  - Apps “sandbox” prevents access to other app’s data
  - System resources, kernel shielded from user apps
  - Inter-app communication only through iOS APIs
  - Code generation prevented
- Mandatory code signing
  - All apps must be signed using an Apple-issued certificate
  - Keychain services
  - Encrypted keychain for storing identities, user names, passwords
- CommonCrypto APIs
- Application data protection
  - Apps can take advantage of built-in hardware encryption

iOS Sandbox

- Limit app’s access to files, preferences, network, other resources
- Each app has own sandbox directory
- Limits consequences of attacks
- Same privileges for each app

Outline

- Introduction
- Apple iOS security model
- Android security model
- Windows 7 Mobile security model

Android

- Platform outline:
  - Linux (Unix operating system) kernel
  - Browser
  - SQL-lite database
  - Software for secure network communication
    - Open SSL, Bouncy Castle crypto API and Java library
    - Java platform for running applications
    - C language infrastructure
    - Bionic LibC (small code, good performance, no GPL)
    - Apache Harmony libraries
    - Also: video stuff, Bluetooth, vibrate phone, etc.
Android challenges

- **Android market**
  - Not reviewed by Google (diff from Apple)
  - Bad applications may show up on market
  - Malware writers may be able to get code onto platform: shifts focus from remote exploit to privilege escalation

- **Battery life**
  - Developers must conserve power
  - Applications store state so they can be stopped (to save power) and restarted - helps with DoS

Security Features

- **Isolation**
  - multi-user Linux operating system
  - each application normally runs as a different user

- **Communication between applications**
  - May share same Linux user ID
    - Access files from each other
  - May share same Linux process and Dalvik VM
  - Communicate through application framework
    - Based on "Binder", discussed in a few slides

Application development process

- **Activity** - one-user task
  - Example: scroll through your inbox
  - Email client comprises many activities

- **Service** - Java daemon that runs in background
  - Example: application that streams an mp3 in background

- **Intents** - asynchronous messaging system
  - Fire an intent to switch from one activity to another
  - Example: email app has inbox, compose activity, viewer activity
  - User click on inbox entry fires an intent to the viewer activity, which then allows user to view that email

- **Content provider**
  - Store and share data using a relational database interface
  - "mailboxes" for messages from other applications

Exploit prevention

- **100 open source libraries + 500 million lines new code**
  - Open source -> no obscurity

- **Goals**
  - Prevent remote attacks
  - Secure drivers, media codecs, new and custom features

- **Overflow prevention**
  - ProPolice stack protection
    - First on the ARM architecture
  - Some heap overflow protections
    - Chunk consolidation in DL malloc (from OpenBSD)

- **No ASLR in initial release**
  - ASLR performance impact
    - Can't install different images on different devices in the factory
  - Later developed and contributed by Bojinov, Boneh
Application sandbox

- Each application runs with its UID in its own Dalvik virtual machine
  - Provides CPU protection, memory protection
  - Authenticated communication protection using Unix domain sockets
  - Only ping, zygote (spawn another process) run as root

- Applications announces permission requirement
  - Create a white list model - user grants access
    - But don’t want to ask user often - all questions asked as install time
  - Inter-component communication reference monitor checks permissions

Layers of security

- Each application executes as its own user identity
- Android middleware has reference monitor that mediates the establishment of inter-component communication (ICC)

Java Sandbox

- Four complementary mechanisms
  - Class loader
    - Separate namespaces for separate class loaders
    - Associates protection domain with each class
  - Verifier and JVM run-time tests
    - NO unchecked casts or other type errors, NO array overflow
    - Preserves private, protected visibility levels
  - Security Manager
    - Called by library functions to decide if request is allowed
    - Uses protection domain associated with code, user policy

Comparison: iOS vs Android

- App approval process
  - Android apps from open app store
  - iOS vendor-controlled store of vetted apps
- Application permissions
  - Android permission based on install-time manifest
  - All iOS apps have same set of “sandbox” privileges
- App programming language
  - Android apps written in Java: no buffer overflow...
  - iOS apps written in Objective-C
- Data protection
  - Android PIN code
  - iOS has delayed lock, answer calls w/o code, other features...
Outline

- Introduction
- Apple iOS security model
- Android security model
- Windows 7 Mobile security model

Windows Phone OS 7.0 security model

- Principles of isolation and least privilege
- Concept of chambers. Each chamber:
  - Provides a security and isolation boundary
  - Is defined and implemented using a policy system
  - The security policy of a chamber defines what operating system (OS) capabilities the processes in that chamber can access

Isolation

Every application on Windows Phone 7 runs in its own isolated chamber
- All applications have basic set of permissions, including a storage file
- Cannot access memory used or data stored by other applications, including the keyboard cache.
- No communication channels between applications, except through the cloud
- Non-MS applications distributed via Windows Phone Marketplace stopped in background
- When user switches applications, previous application is shut down
- Reason: application cannot use critical resources or communicate with Internet-based services while the user is not using the application

Four chamber types

- Three chamber types have fixed permission sets
- Fourth chamber type is capabilities-driven.
  - Applications that are designated to run in the fourth chamber type have capability requirements that are honored at installation and at run-time.

Overview of four chambers

- **Trusted Computing Base (TCB) chamber**
  - unrestricted access to most resources
  - can modify policy and enforce the security model
  - kernel and kernel-mode drivers run in the TCB
  - Minimizing the amount of software that runs in the TCB is essential for minimizing the Windows Phone 7 attack surface

- **Elevated Rights Chamber (ERC)** can access all resources except security policy.
  - Intended for services and user-mode drivers

- **Standard Rights Chamber (SRC)** is default for pre-installed applications.
  - All applications that do not provide device-wide services
  - Outlook Mobile is an example that runs in the SRC

- **Least Privileged Chamber (LPC)** is default chamber for all non-Microsoft applications
  - LPCs are configured using capabilities as described in the following slide.

Source: Windows® Phone 7 security model, OEG331 12/2010
Windows Phone 7 “Capabilities”

- W7 Capability: a resource associated with user privacy, security, cost, or business concerns.
- Examples: geographical location information, camera, microphone, networking, and sensors.
- The LPC defines a minimal set of access rights by default, and can be expanded using capabilities.
- Capabilities are granted during application installation, and their privileges cannot be elevated at run time.

Granting privileges to applications

- Goal: application receives all capabilities that are needed to perform all its use cases, but no more.
- Each application discloses its capabilities to the user:
  - Listed on Windows Phone Marketplace.
  - Explicit prompt upon application purchase, for those capabilities that have legal requirements for explicit disclosure and specific consent collection.
  - Disclosure within the application, when the user is about to use the location capability for the first time.
- Developers use the capability detection tool that is distributed with the Windows Phone Developer Tools to create the capability list for their application.
- The capability list is included in the application manifest in the application package (WMAppManifest.xml).

Managed code

- Application development model uses of managed code only.

.NET Code Access Security

- Default Security Policy is part of the .NET Framework:
  - Default permission for code access to protected resources.
  - Permissions can limit access to system resources:
    - Use EnvironmentPermission class for environment variables access permission.
    - The constructor defines the level of permission (read, write,...)
  - Deny and Revert:
    - The Deny method of the permission class denies access to the associated resource.
    - The RevertDeny method will cause the effects of any previous Deny to be cancelled.

Example: code requires permission

```csharp
class NativeMethods
{

    // This is a call to unmanaged code. Executing this method
    // requires the UnmanagedCode security permission. Without
    // this permission, an attempt to call this method will throw a
    // SecurityException:
    [DllImport("msvcrt.dll")]
    public static extern int puts(string str);

    [DllImport("msvcrt.dll")]
    internal static extern int _flushall();
}
```

Example: Code denies permission not needed

```csharp
private static void MethodToDoSomething()
{
    try
    {
        Console.WriteLine("... ");
        SomeOtherClass.method();
    }
    catch (SecurityException)
    {
        ...
    }
}
```
.NET Stackwalk

- Demand must be satisfied by all callers
  - Ensures all code in causal chain is authorized
  - Cannot exploit other code with more privilege

Stackwalk: Assert

- The Assert method can be used to limit the scope of the stack walk
  - Processing overhead decreased
  - May inadvertently result in weakened security

Summary

- Introduction
- Apple iOS security model
- Android security model
- Windows 7 Mobile security model

Differences between platforms

- Operating system
  - Unix
  - Windows
- Approval process for applications
  - Market: Vendor controlled/Open
  - App signing: Vendor-issued/self-signed
- Programming language for applications
  - Managed execution: Java, .Net
  - Native execution: Objective C