HOCHSCHULE DER MEDIEN

# Mobile Malware Evolution and the Android Security Model

Roland Schmitz

Hochschule der Medien Stuttgart

schmitz@hdm-stuttgart.de

# Where do I come from?

# Study Programs in the Computer Science and Media Department

- Computer Science and Media (B. Sc.)
  - 6 semester course
  - 10 professors
  - Provide a solid education in computer science with applications to media technology
- Mobile Media (B. Sc.)
  - newly established 7 semester course
  - 3 professors (yet to be called), close cooperation with computer science and media
  - Provide an interdisciplinary education in the field of mobile media, with a strong technological background
- Computer Science and Media (M. Sc.)
  - Well established 4 semester master course
  - Qualify students for project leader or management positions
- More info: www.mi.hdm-stuttgart.de

# Agenda

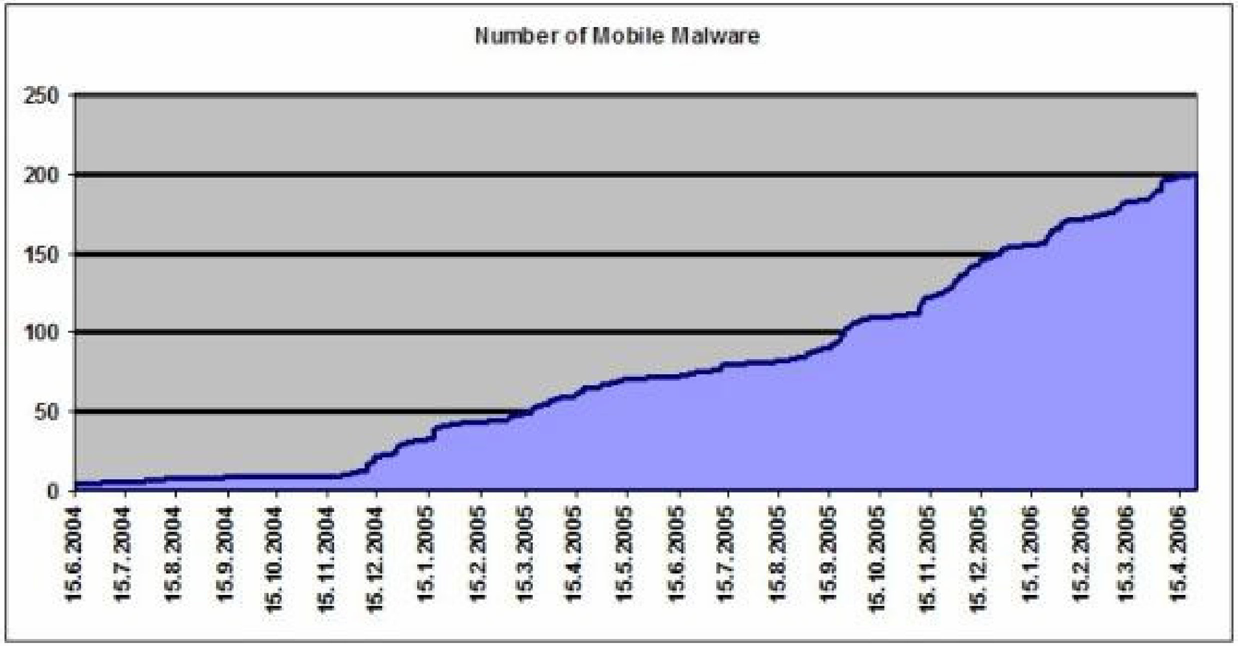
- Mobile Malware
  - Motivation
  - Facts and Figures
  - Some History
  - The Android Browser Bug
- Android Security Model
  - Overview
  - Key Features and Pitfalls
  - Evaluation
- The Future?

# Why Mobile Malware?

- Growing complexity of smartphones makes them more vulnerable than in the past
- Often users are not aware of any danger
- Sensitive data stored on Smartphones
- „Always-On“ makes spreading of malware easier
- User tracking possibility, e.g by using GPS coordinates
- Financial Motivation
  - Mobile Banking
  - Mobile Payment
  - Premium-Service Numbers

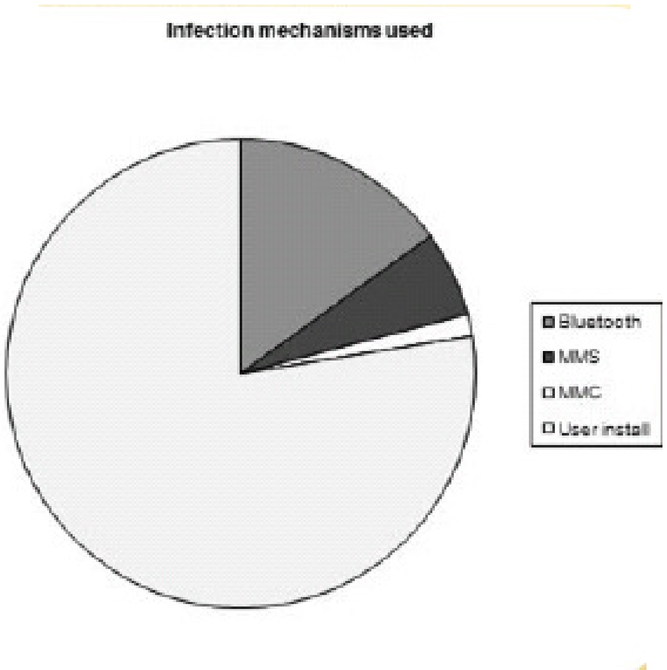
# Growing Number of Reported Mobile Malware (until 2006)

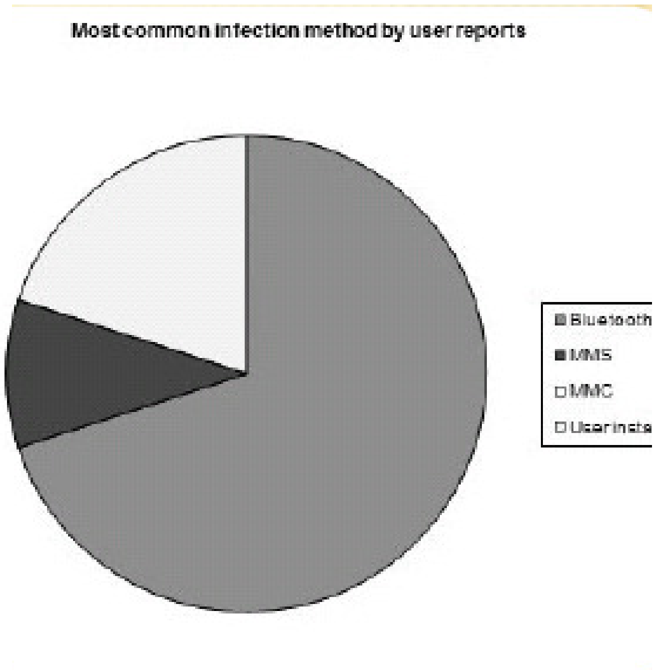
HOCHSCHULE DER MEDIEN



Source: F-Secure.com

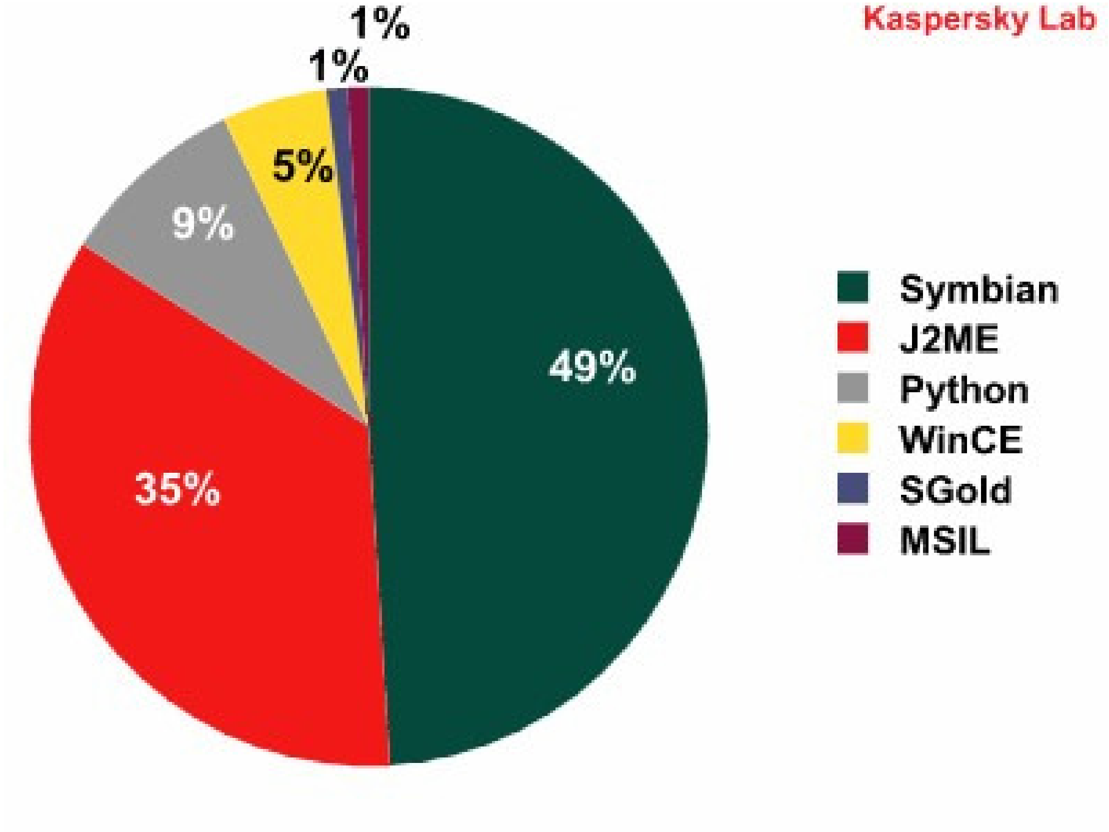
# Mobile Malware
# Infection Mechanisms





- User install and bluetooth are by far the most important infection mechanisms
- Infection via bluetooth shows same spreading pattern as biological viruses

# Affected Platforms (by 6/2009)



# Mobile Malware: The Beginnings

- June 2004: Worm.SymbOS.Cabir.A
  - First reported mobile malware
  - „Proof of concept“
  - Spreads via bluetooth, user has to download and execute code
- July 2004: Virus.WinCE.Duts
  - First virus written for Windows Mobile
  - Infects exe-files
  - Needs user approval for infection
- November 2004: Trojan.SymbOS.Skuller
  - Replaces program icons with skulls
  - Infection via „warzed installers“
  - Uses security hole in Symbian

# Mobile Malware:
# Getting serious

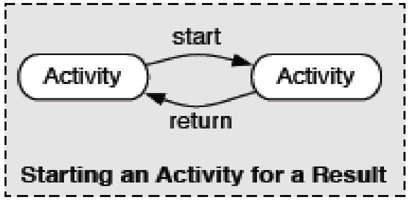
- March 2006: Trojan-Spy.SymbOS.Flexispy
    - Collects information about calls and SMS
    - First example of mobile spyware
- May 2007: SymbOS.Viver.A
    - Sends MMS to premium service numbers
    - First example of mobile malware with explicit financial background
- January 2008:Trojan.iPhone.A
    - First reported malware for iPhone
    - Replaces legitimate applications
- October 2008: First Android Phones commercially available
    - The same month, a first vulnerability is reported…

# The Android Browser Bug

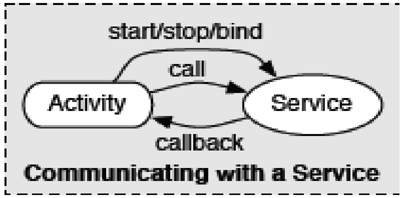
- Identified and exploited by Charles Miller, Mark Daniel and Jake Honoroff of Independent Security Evaluators in October 2008
- If a user visits a malicious site, the attacker can run any code *with the privileges of the web browser application*.
- Thus, the impact of the attack is limited to data the browser has access to:
  - Cookies
  - Saved passwords
  - Information put into web applications

# Android Component Model

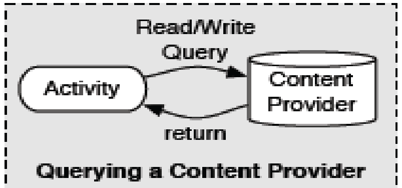
- Each application runs as its own UNIX uid
- Sharing can occur through application-level interactions
- Interactions are based on components. Different component types are:
  - Activity
  - Service
  - Content Provider
  - Broadcast Receiver



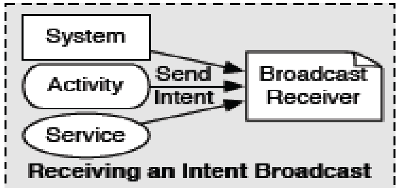
Starting an Activity for a Result



Communicating with a Service



Querying a Content Provider

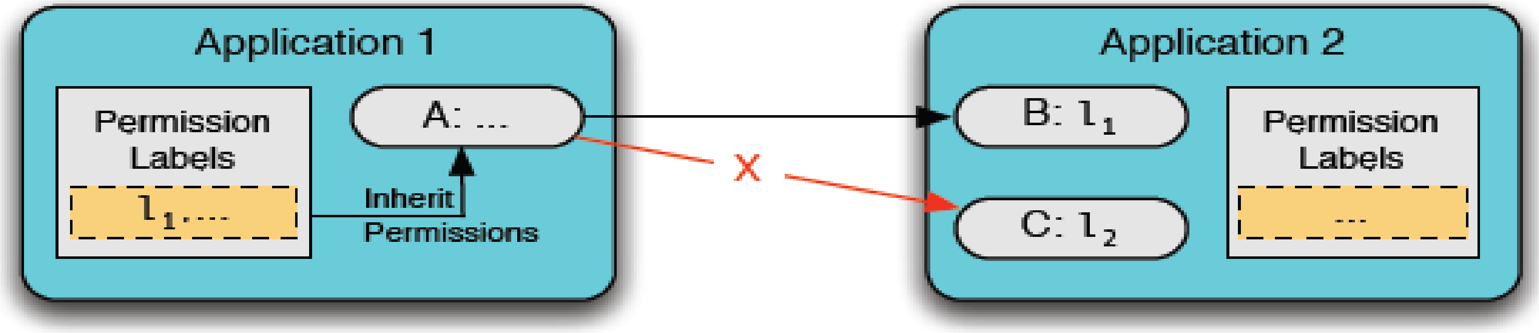


Receiving an Intent Broadcast

- Target components may be in the same or different applications

# Android Security Model Overview

- Android focuses on Inter Component Communication (ICC)
- The Android manifest file allows developers to define an access control policy for access to components
  - Each component can be assigned an access *permission label*
  - Each application requests a list of permission labels (fixed at install)
- Android's security model boils down to the following picture:



# Android Security Key Features

- Isolation
  - Each application runs as its own uid
  - uid sharing only if developer‘s signature keys are the same
- Code Signing
  - Each application must be digitally signed
  - Self-signed certificates are possible
- Mandatory Access Control
  - Developers may define access control rules to their components
  - Sensitive system resources are protected by permissions
- Permissions are statically assigned at install time
  - *Normal* permissions are assigned per default
  - *Dangerous* permissions are granted by user
  - *Signature* permissions are granted only to applications signed by the same developer key

# Android Security Evaluation

- ⊕ Isolation by different uids per application is a major step towards limiting potential damages
- ⊕ Basic MAC model is easy to understand
- ⊕ Network and hardware resources are protected by permissions
  - Applications must request these permissions in their manifest
  - Makes it easier to evaluate an application‘s security
- ⊖ Non-trivial security decisions are left to the user
- ⊖ Possibility to delegate actions via *Pending Intents* may cause problems („Confused Deputy Problem“)
- ⊖ Code-Signing might lead to a false feeling of trust at the user‘s side

# The Future?

- Android will become a major target for malware authors
- Mobile Anti-Virus Solutions are already available
- Android security model seems to be better designed than competing operating systems
- Developers must know and implement the security model at code level
  - currently focus is on platform version updates and features.
- Users need to be informed about security risks and the possible impact of granting access permissions
- If possible, users should be relieved from having to take critical security decisions

# Thanks for your attention!

Do you have any questions?