Disambiguation

• Topic: “Viruses, Worms and Trojans on mobile phones”

• Viruses, Worms and Trojans
  – Neglect differences → “malware”

• Mobile phones
  – Modern smartphones (OS and apps)
  – But:
    • Tablets, Pocket PCs
    • Feature phones (Java)
Table of contents

- History of “mobile malware”
- Goals of malware
- Attack vectors
- Attack analysis
  - Low-level attacks
  - High-level attacks
- Prevention of infection
- Outlook
History of “mobile malware” I

- Cabir: First malware “in the wild”
- June 2004
- Worm for Symbian
- Spread via BlueTooth, sends installer to next best discoverable device:
  - Recipient has to confirm manually
- Proof-of-concept, no harm done
History of “mobile malware” II

• But:
  – BlueTooth distribution revolutionary, inspired future malware writers
  – Source code of Cabir was published → new dangerous variants
• In 2005 “CommWarrior” spread additionally via MMS
History of “mobile malware” III

• 2004-2011
  – New mobile phone operating systems (Android, iOS, WebOS, Windows Phone...)
  – Increasing distribution of J2ME
    • Sand-boxed execution, harming phone is difficult
    • **But:** allows for sending SMS to premium rate numbers, **cross-platform** SMS trojans
History of “mobile malware” IV

• Summary (2004-2011):
  – Spread via 3rd party app-store & WAP portals
    • Trojans with direct financial gain (premium rate numbers, e.g. J2ME SMS trojans)
    • Rendering device useless (e.g. Skuller)
  – Very few powerful attacks with goal to steal data:
    • Proof-of-concept implementations by researches (bugging/spying on the phone)
    • Only few successful black-hat distributions so far (e.g. WinCE.InfoJack, Android Geinimi, SymbOS.Yxe, iOS firmware v1.1.3...)
Goals of malware

- Depends on author
  - White-hat hackers (researchers): Develop proof-of-concept implementation
  - Raise attention, (OS programmers, AV specialists)
  - Usually spread via 3rd party app stores
- Black-hat hackers: earn money, execute power, or simply harm the victim
Goals of malware II

- Earn money
  - Call/text premium rate numbers
  - Steal banking data
  - Harvest data, sell to 3rd parties (enterprise information!)

- Exercising power
  - Trojan horse with back-door (remotely control or bug the phone, e.g. tap phone calls, text messages, GPS location...)

- Harming the user...
Attack vectors

Malware as payload
of non-executable data, e.g.
Adobe Flash, PDF, Webkit (HTML)

Low-level attacks
1) on particular communication stacks
   (e.g. sending illegal frames)
2) on software components/services
   (e.g. NetBIOS service, SMS parser...)
   Unattended infection!

Installed applications
Applications deliberately
installed by the user:
1) from app store
2) Recommendation
   from friends (via SMS,
   MMS, email, FB...)
3) BlueTooth

Physical access
Installation of malware while
user is absent (e.g. Mobile-Spy)

USB - PC connection
1) Act as carrier (thumb-drive) for
   PC-to-PC malware
2) Infection by PC-malware
Attack vectors: USB connection I

- Phone (SD card) mounted as USB thumb-drive
  - PC ↔ PC infection: Allows infected computer to spread malware to other computers
    • Auto-run no longer works → manual execution
  - Phone → PC infection: Infected phone, place PC malware on SD card (e.g. inject into existing binaries)
  - PC → phone infection: Infected computer identifies the phone and places malware on it
Attack vectors: USB connection II

- Phone (SD card) mounted as USB thumb-drive (continued)
  - PC → phone infection (continued):
    - Identification of the exact phone model is possible with some models
    - Idea: guess phone's OS, place malicious file on SD card, will be processed by phone's application (e.g. gallery thumbnail)

Exact serial number

- Infect phone via synchronization mechanisms
  - “Crossover” (02/2006) spread from Windows PC to Windows Mobile/CE phone via ActiveSync
Attack vectors: physical access

- Gain physical access to someone's phone in their absence, install spyware (e.g. via download)
- There actually is legally purchasable spyware:
  - Used by “concerned” husbands, parents, employers
### Attack vectors

- **Malware as payload**
  - of non-executable data, e.g.
    - Adobe Flash, PDF, Webkit (HTML)

- **Low-level attacks**
  1) on particular communication stacks
     - (e.g. sending illegal frames)
  2) on software components/services
     - (e.g. NetBIOS service, SMS parser...)
     - *Unattended* infection!

- **Installed applications**
  - Applications deliberately installed by the user:
    1) from app store
    2) Recommendation from friends (via SMS, MMS, email, FB...)
    3) BlueTooth

- **Physical access**
  - Installation of malware while user is absent (e.g. Mobile-Spy)

- **USB - PC connection**
  1) Act as carrier (thumb-drive) for PC-to-PC malware
  2) Infection by PC-malware
Attack vectors: low-level attacks I

- Sending/receiving data over 3G, WiFi, BlueTooth, radio controlled via drivers (*software stack*)
  - Vulnerabilities can be found via “fuzzing”
    - Send random data *frames* to the stack, check for crashes of the targeted process, examine for possibility of code execution
  - Drivers run in kernel space → *jackpot* if security issues are found
Attack vectors: low-level attacks II

- On the next higher level:
  - Attack components in charge of SMS and MMS processing
  - Target permanently running services, e.g. naming services (NetBIOS, Bonjour)

- All this is not just theory!
  - Successful attacks on SMS and MMS stacks already illustrated
  - Bluesnarfing (attack on BlueTooth stack)
Attack vectors: high-level attacks

- High-level attack = user installs application
  - Source: 3rd party or official app stores
  - Apps claim to have a valid purpose (banking application, game...), but come with a malicious component → “trojan horse”
  - Installation by own initiative or (fake) recommendations from friends → higher level of trust
Attack analysis

• Now: analysis (advantages, disadvantages) of
  – Low-level attacks
    • Stack implementations
    • Malware as payload
    • PC → Phone via USB
  – High-level attacks
    • Installing apps
    • (Physical access)
Attack analysis: low-level attacks

- Advantage: unattended infection
- Disadvantage: efforts for the malware developer:
  - Malware developed in two stages
    - Stage 1: Develop functionality (high-level, C) and the machine code to be injected into buffer (time consumption fixed, 13 weeks, fulltime)
    - Stage 2: Find an application with buffer-overflow vulnerability (time consumption variable)
Spreading of low-level attacks

- Unattended infection ↔ real-world virus infection (e.g. influenza)
- Outbreaks (Windows), e.g. Nimda or ILoveYou
  - And in the world of mobile phones? Nada! Why?!
- Examine spreading pattern considering the technologies, MMS and BlueTooth
Spreading using BlueTooth I

- Infects devices in its proximity (as does a real virus) → “proximity malware”
- P. Wang et al, "Understanding the spreading patterns of mobile phone viruses" (04/2010) conducted experiments, using a mix of mathematical mobility models and collected mobility data from cellphone-towers
- BlueTooth malware eventually reaches all susceptible devices
  - Higher market share → faster spread
Spreading using BlueTooth II

• Exemplary:
  - \( m = 0.3 \) (30%) \( \rightarrow \) 85% of devices infected in a few hours
  - \( m = 0.01 \) (1%) \( \rightarrow \) completion of infection takes several months

• Consider Symbian (at the time \( m_{symb} = 0.643 \))
  - But: \( m_{symb} \) is the relative market share: multiply with \( m_s = \) market share of smartphones = 0.05
  - \( m_{symb} \times m_s = 0.032 \) (3.2%)
Spreading using MMS messages I

- Sending MMS messages → proximity-independent
- P. Wang et all illustrate, spreading using MMS
  - Depends on $m$
  - Depends on underlying graph representing connections between users
"Percolation transition point" $m_c \approx 0.095$ (9.5%)
Spreading patterns: result

- Smartphones still the minority on the phone market
- But: Smartphones are a new phenomenon
  - Sales figures are constantly climbing (Gartner, 08/2010)
    - This will grow even stronger, as Eric Schmidt (Google) suggests: “We want to increase the availability of inexpensive smartphones in the poorest parts of the world. We envision literally a billion people getting inexpensive, browser-based touchscreen phones over the next few years”

<table>
<thead>
<tr>
<th></th>
<th>Units sold (Q2/09)</th>
<th>Units sold (Q2/10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>286'122'300</td>
<td>325'556'800</td>
</tr>
</tbody>
</table>
Attack analysis: high-level attacks

- **Advantage:** malware developer can focus on functionality (use high-level API)
- **Disadvantage:**
  - Manual installation by the user, grant requested access rights
    - But: User will assume apps from the *official* store are *safe*
  - Costs incurred to publish app in official store
    - But: Costs usually low compared to earnings (even with Apple's app-store fee of $99)
High-level attacks: Trustworthiness

- Q: Trust in the safety of official apps justified?
- Apple performs static and dynamic code analyses (can be tricked)
- Other parties (Google, RIM...) sell “official” keys to developers, sign and publish apps
  - Incurred fees ($0 - $25) are *nothing* compared to potential earnings
High-level attacks: Security models

- Mobile phone OSes come with a “security model”
  - User controls HW/SW components the app may access
- Problems:
  - User swamped with technical concepts and terms
    - Becomes overburdened, loses interest
  - Access rights insufficient for decision (e.g. movie player vs SMS tool)
  - User is “trained” to accept any requested access
User permissions: Example

• Looks like a package insert of your favorite drug... Who reads that again?

• Result: TrendMicro survey: “Only 23% of the users use the security features...”
Prevention of infection I

- Operator of the **app-store**: introduction of static and dynamic analyses to reduce chance of malware being published
- Programmer of **OS**: tighter default values for security framework
- **Network provider**: collaborate with AV specialists to minimize infection over provider-controlled channels (SMS, MMS, 3G internet)
Prevention of infection II

**User:**
- Use anti-virus software
- Disable BlueTooth (avoid proximity malware)
- Use locking mechanisms (prevent 3rd party infecting the phone physically)

**Society:** Advertise the threat of malware to mobile phones. Apply knowledge from the PC world to the mobile phone
Outlook

• High-level attacks in near future
  – User-base will keep ignoring the risk, until enough severe incidents have happened
• Market share of smartphones will rise, attacks will become increasingly worthwhile
  – Outbreaks like Nimda? Predominant platform?
• Low-level attacks: a niche-product until war of mobile operating systems has settled?
Thanks for listening

The End
Supplementary material
Comparison: PC vs. mobile phone

- Malware attacks are well-known from the PC Windows world
  - Symantec Antivirus detected 1'122'311 malicious programs in 04/2008

- For mobile phones (Kaspersky, 09/2009):

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Number of variants</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbian</td>
<td>253</td>
</tr>
<tr>
<td>J2ME (Java Microed.)</td>
<td>182</td>
</tr>
<tr>
<td>WinCE/Mobile</td>
<td>26</td>
</tr>
<tr>
<td>Others</td>
<td>54</td>
</tr>
</tbody>
</table>

- Doesn't even reach 1'000
History of “mobile malware” eI

- **Skuller**, detected in 11/2004
  - Cabir's spread method
  - Exploited Symbian OS vulnerability
  - Removed application executables and replaced icons with skulls
  - In disguise as “Extended theme”:
History of “mobile malware” eII

- **CommWarrior**, created in 03/2005
  - Spread via BlueTooth, to **all** discoverable phones
  - **Additionally** spread via MMS (payload 30 KB)
  - Over 20 interesting message titles, such as “3DNow! 3DNow!(tm) mobile emulator for *GAMES*”, or “WWW Cracker Helps to *CRACK* WWW sites like hotmail.com”
    - Safe-guard to not affect other insiders(?)
  - No purpose other than spreading → classify as **worm**