PCTEL SECURE

Smartphone Security Architectures

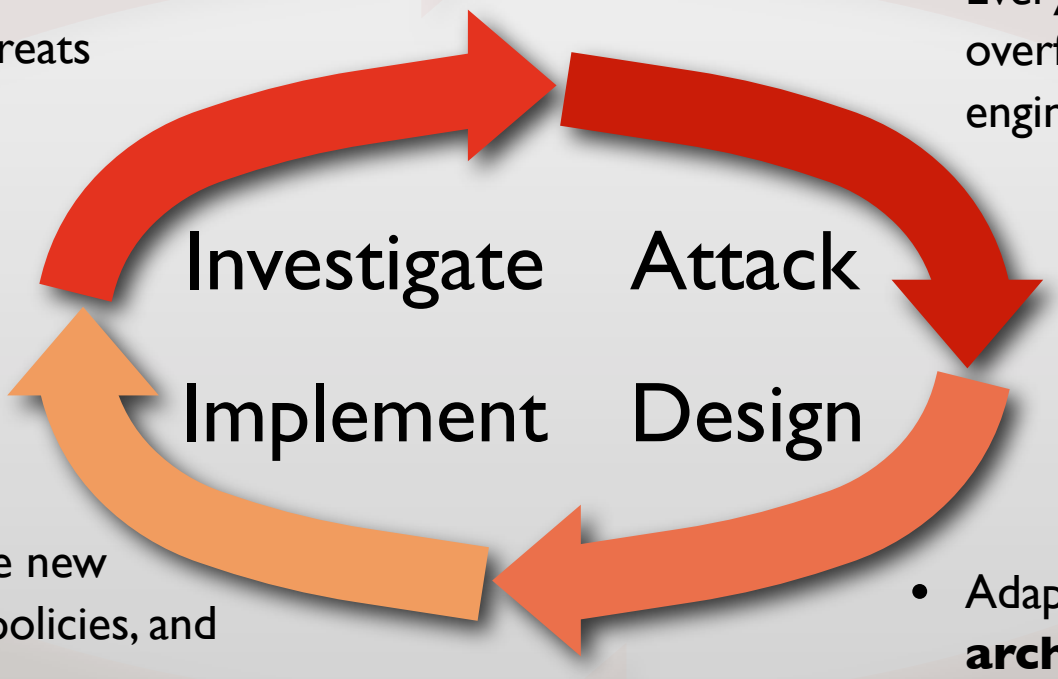
Eric Üner

ACSAC 2011

Typical modus operandi

- Investigate known attacks
- Research solution
- Determine threats

- Test existing attacks
- Develop new attacks
- Everything from buffer overflows to social engineering

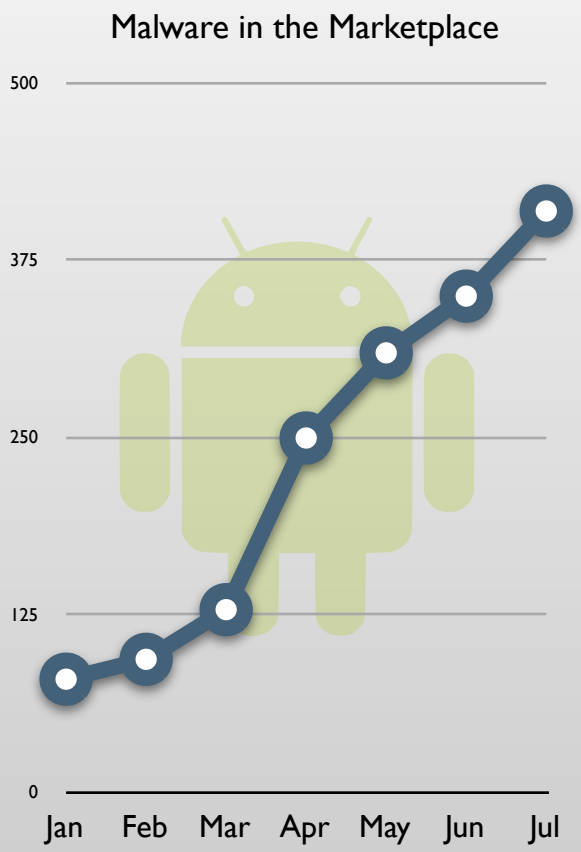


- Implement the new architecture, policies, and designs
- Fix the any known low-level security defects that remain
- And since you can never win, start again...

- Adapt or change the **architecture**
- Adapt or change the **environment**
- Determine what we can do at the low-level (in the code)

Challenge: Android

- Android malware affected over one million users in 2011
- Android trojans found in numerous apps on Android market, Google struggling to keep malware off of Android phones
- One third of all Android owners are likely to encounter threats on their device this year
- Malware records voice, intercepts emails, and more, and is not stopped by sandboxing, encryption or anti-virus tools
- We need to see if this is really an issue, or just marketing...



Sources: eWeek, [in]Secure, WSJ, IBT

So we made some malware

Motorola Droid
Android 2.1 (Eclair) Linux 2.6.32.9
(Custom version)

Samsung Intercept
Android 2.1 (Eclair) Linux 2.6.29

HTC Incredible
Android 2.2 (Froyo) Linux 2.6.32.17

Motorola Droid X
Android 2.3.3 (Gingerbread) Linux 2.6.32.9

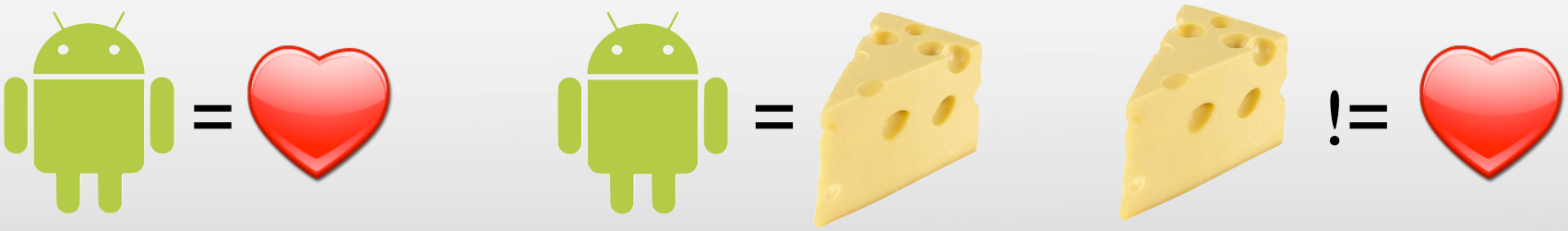
LG Optimus V
Android 2.2 (Froyo) Linux 2.6.32.17

Motorola Atrix
Android 2.3.4 (Gingerbread) Linux 2.6.32.9

- Attacked a wide variety of devices
- Infection/delivery uses a variety of exploits and attacks
  - Webkit bugs
  - Linux exploits
  - Trojans

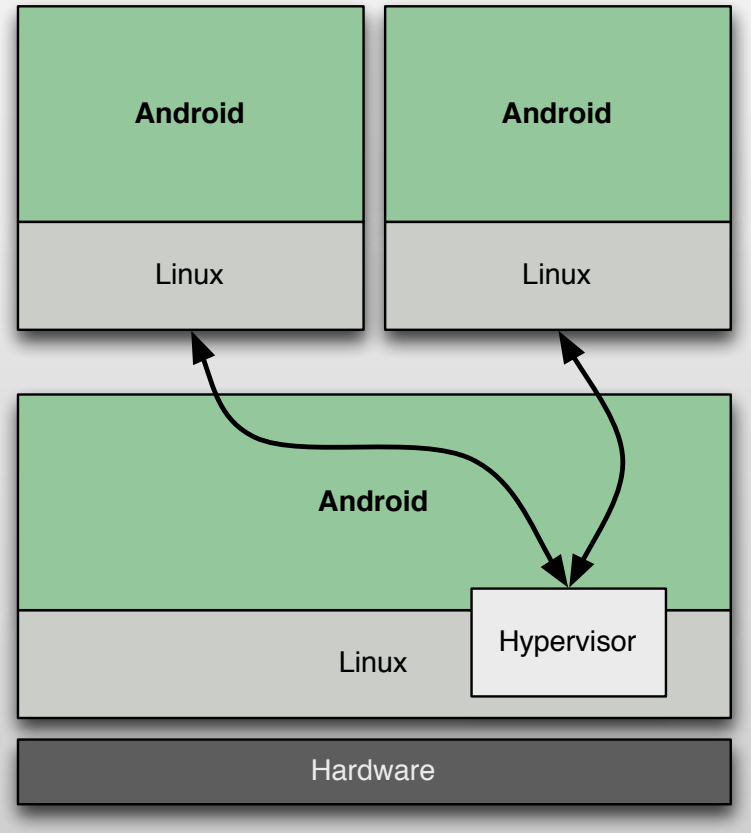
- Steals data
  - Downloads
  - Photos
  - Cache and other data
- Enables mic and records audio
  - Turns phone into a bug
- Uploads it all to a server

How do we fix it?



- Android is big and complex, and will **always** have bugs and weaknesses
- It is a general-purpose OS, and we want to keep its flexibility
- We can't change the environment (users, user cases, network, etc.)
- Three approaches to the architecture:
  - Virtualization and Type 2 hypervisors
  - Type 1 hypervisors (microvisors, RTOSs, bare-metal hypervisors)
  - An interesting alternative

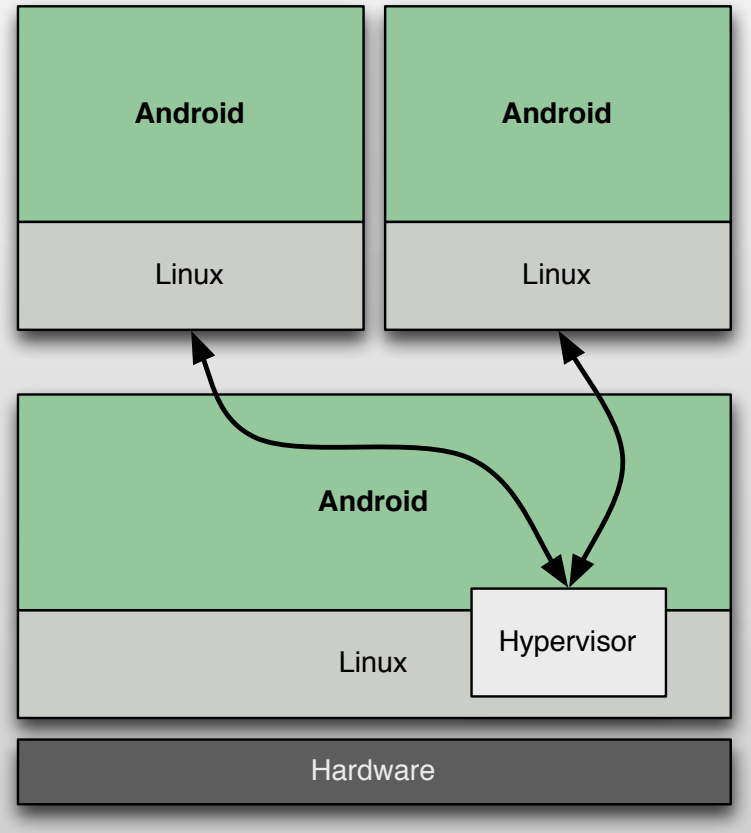
Type 2 hypervisor/virtualization



Advantages

- Can run multiple operating systems besides Android (e.g. Windows Mobile and Android)
- Perfect for enterprises that want to have multiple configurations
- May be able to take advantage of hardware features to assist virtualization
- Could run versions of Android or other OSs that are not designed for the physical hardware

Type 2 hypervisor/virtualization

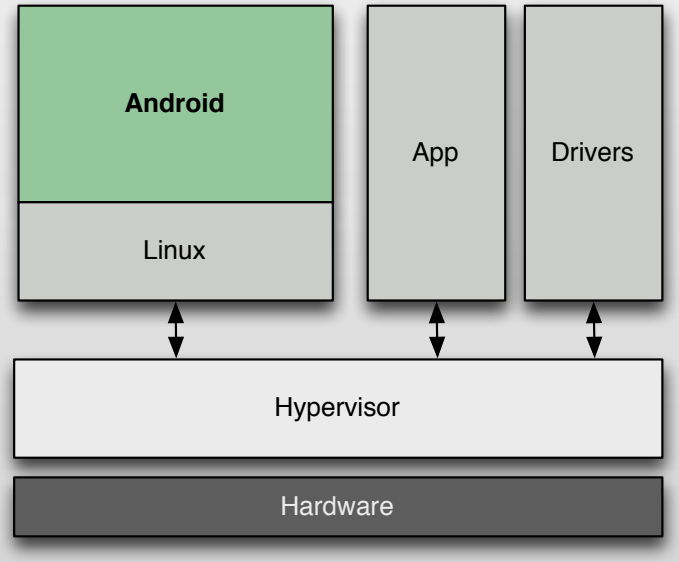


Problems

- Security posture depends on host OS (i.e. Can never be more secure than Android is normally)
- Guest OSs are not strongly separated
- No actual security for apps or the OSs
- Performance degradation

[Malware still effective]

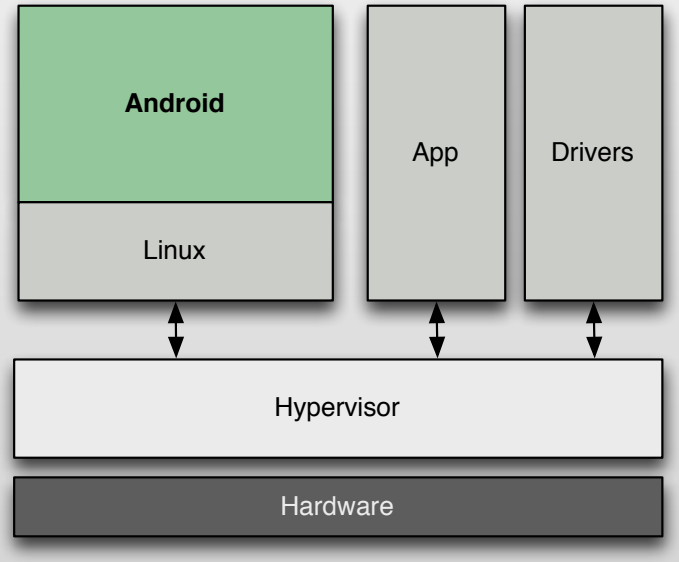
Type 1 Hypervisor/RTOS/bare-metal



Advantages

- Strong separation between apps and drivers placed in their own cells and the rest of the OS
- Could offer better performance than Type 2
- Smaller trusted computing base

Type 1 Hypervisor/RTOS/bare-metal



Problems

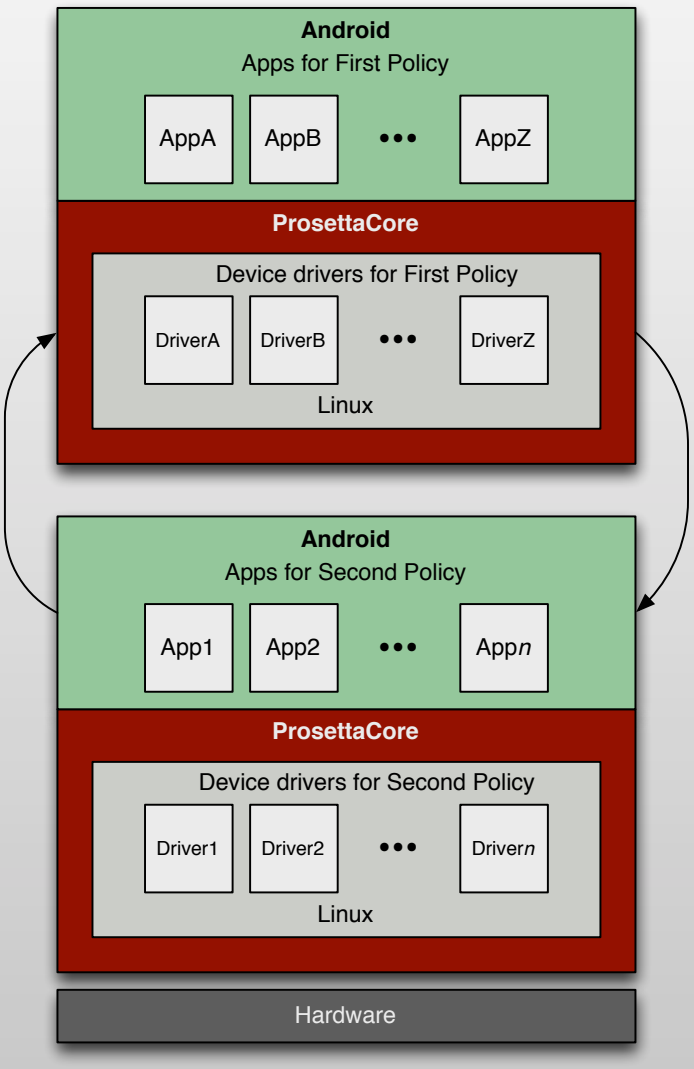
- Have to move specific apps and drivers to defend against specific threats
- The more you defend against, the larger your trusted computing base, negating a key advantage
- Intense engineering effort
- Requires tight hardware coupling
- Still no security for the Android apps

[Malware still effective]

So how to stop the malware?

- Sandboxing techniques did not stop it
- SELinux had complexity issues
- We looked at dozen of techniques and waded through commercial and academic alternatives
- We needed something simpler, that works on a wide variety of threats on a wide variety of platforms

The approach we chose



- Multiple security policies create multiple Androids, where only one may be active at a time
- Thin layer around Linux provides monitoring, policy enforcement, and integrity checking
- Some key enhancements to Linux to add MAC and other security features

Advantages

- Minimal engineering effort
- Strong separation (especially between apps that are not even running)
- The best of both worlds

Malware thwarted!

Solution overview

- Phone is flashed with trusted, cryptographically signed firmware from PCTEL Secure
- Security policies managed on a policy servers decide precisely what the phone can do, where, and when
- Multiple, distinct, isolated security domains are now possible on a single Android smartphone, each with their own capabilities, files, encryption, networks, etc.

Key advantages with this architecture

- Behavioral vs taxonomic analysis
  - Not looking for specific threats, viruses, files, or patterns, but rather any behavior not allowed in the policy
- Policies
  - Temporal in addition to cryptographic and other isolation methods
  - Sandboxing, virtualization, etc. cannot provide this level of isolation
  - You cannot attack something that is not there
- Biomorphics
  - An attack on one device will not work on a different device, or even the same device later in time
- Device support
  - Moves easily to new devices without the re-engineering efforts involved in porting virtualization or hypervisor solutions

What you can do

- Analyze your requirements
  - Decide how important security is to your solution (if it's not important, repeat this step until it is)
  - Determine if you will have a specialized, single device
  - Analyze the diversity of the adversaries and assets to protect
- Let us know how we can help

Thank you

PCTEL SECURE

http://www.PCTELSecure.com
Eric Uner