Mythbusters

- Security means different things to different people
- Closed source more secure than open source
- Security could be achieved by obscurity
- Software-only security is good [enough]
- Security folks are pain in the *neck*
- Security is a set of components
- Can protect against all attacks
- Encryption equals security
- Can add security later on
- Hackers are clueless

Android: A Security Analysis

Hadi Nahari
Chief Security Architect
NVIDIA

"Veni ad Android sepeliendum, non ad laudandum" –Bill ShakesP2P

NVIDIA

Hadi's ego slide

- Security, Cryptography, Complex System Analysis ID Management, Asset Protection, Information Assurance Schemes
- Massively Scalable Systems design, implementation, and governance, Vulnerability Assessment, Threat Analysis (VATA)
- Theory of Programming Languages, Formal Languages, Functional Languages, Semantics of Security
- Enterprise & Embedded (Netscape, Sun Microsystems, United States Government, Motorola, eBay, PayPal, NVIDIA…)
- Author of "Web Commerce Security: Design and Development" book, published by John Wiley & Sons

Agenda

- Security
- Android Security
- Case Studies
- Conclusion

Agenda

- Security
- Android Security
- Case Studies
- Conclusion

Security is not rocket-surgery

- Security is defined by two things
  - Assets: what you protect
  - Threats: what to protect against
- Without assets and threats, security is meaningless
- Security is a subset of QA/verification
  - "exclusive specification verification" definition
- Security is hard to measure (very hard)
  - State-space combinatorial explosion: cannot enumerate all attacks
  - The secure state of a complex system is "practically" undefinable

HW vs. SW: does it matter?

- A SW-only security solution is prone to system-attacks
  - Well, almost always
- HW is a good base to address system-wide attacks
  - Well, most of the time
- In either case, common principles apply
  - Authentication: strong, mutual, verifiable, “frequent”
  - Authorization: mandatory, abstracted, enforced, chained
  - Public vs. private key material (both should be tamper-proof)
- Two requirements to satisfy
  - ROT
  - COT

Root Of Trust (ROT)

- Root of trust is the lowest verified-entity in a [security] call stack
- It relies on verification of identity
  - Difference between Identification & authentication
- Root of trust could be in software or hardware
  - Where else?
- Interview question:
  - What is the root of trust in an SSL communication?

Chain Of Trust (COT)

- It's not sufficient to have a solid ROT
- Multiple system entities participate in actions
- Passing control from one entity to another: attack entry
- Interview questions
  - What is SecureBoot (or HAB: high assurance boot)?
  - What problem does it solve?
  - What problem does it not solve?

Cryptography: is it, like, photography??

- No, it's not (just in case you were wondering)
  - Although steganography is close to both
- The mathematical systems for securing
  - Communication or DIT (data in transit)
  - Storage or DAR (data at rest)
- Getting the cryptography right is hard. Really, really hard
  - The Kirchhoff's principle
- Cryptography is the easiest part of securing your system
- Trivia question
  - Why was the term "decryption" banned in middle ages?

Ring_0: got TEE?

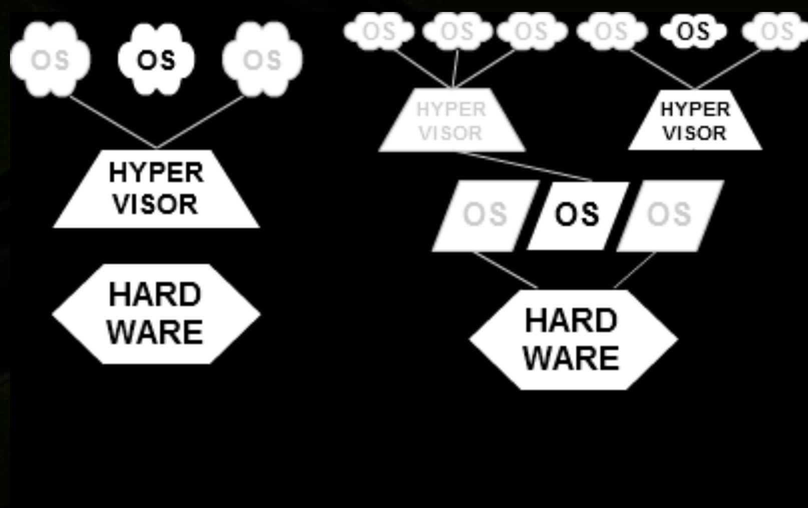
- CS101: a modern OS usually has four rings
  - Rings are logical representation of access control (got Authorization?)
  - Ring_0 entities have the highest system privileges
- A modern OS is a very complex spaghetti of modules, components, devices, subsystems, etc.
- Even “defining the security of such a mess brings tears to grown-up eyes
  - Let alone implementing, proving, and verifying it…

Ring_0: got TEE? *(cont'd)*

- There are two ways to address the problem of complex call stack (from security POV)
  - Verify every single call and execution path to trusted process/app (the classical "CIA agent in Moscow" problem)
  - Implement what's called "secure isolation"
- Separation/segregation vs. secure isolation
- TEE (trusted execution environment) and TCB (trusted computing base)
  - Both HW & SW flavors exist
  - TEE implementations have a S/W stack to support H/W
  - The stack is either a hypervisor, a monitor, or a microkernel
  - [almost] all implementations exist

Hypervisor

- We use Hypervisor <u>equivalent</u> to VMM
  - Virtual machine manager
  - Virtual machine monitor
- virtual machine(s), monitor, guest OS's: two types



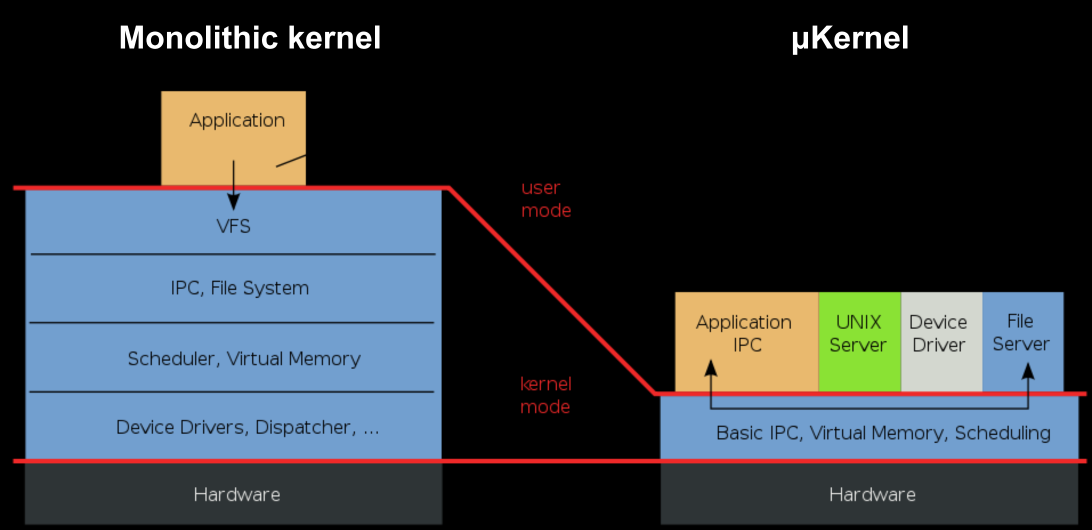
Type I — **(native/bare metal)**

Type II — **Hosted**

(picture courtesy Wikipedia)

µKernel

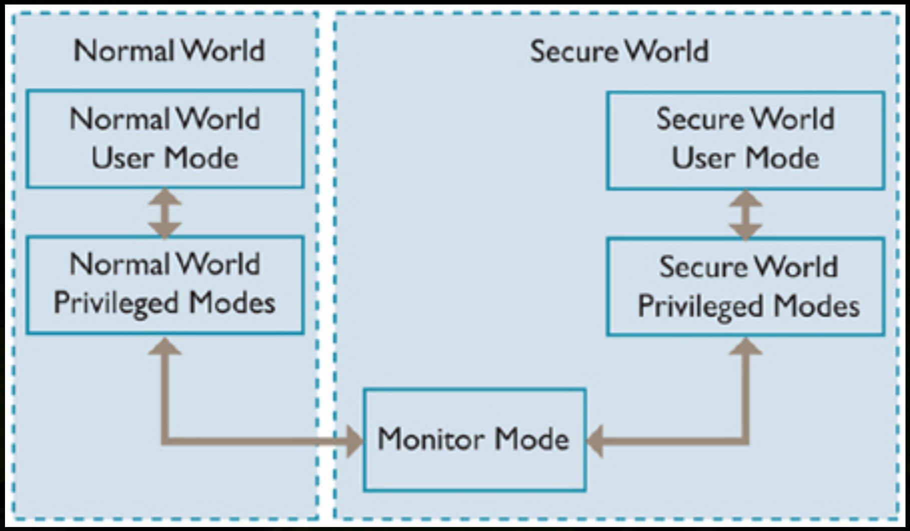
- As opposed to monolithic
- It's just that: a minimum kernel of an OS
- That is, minimize the crap (technical term) in the Ring_0
- Push everything northbound to user space
    - Why is it a good idea from security POV?



(picture courtesy Wikipedia)

Typical TEE Implementation (TrustZone)

- It's a combo approach



(picture courtesy ARM)

Agenda

- Security
- **Android Security**
- Case Studies
- Conclusion

Android: what is it?

- Linux-based software stack for “mobile” devices
- Very divergent from typical Linux
- Almost everything above the kernel is different
  - Dalvik VM, application frameworks
  - bionic C lib, system daemons
  - init, ueventd
- Heck: even the kernel is different
  - Unique subsystems/drivers: Binder, Ashmem, …
  - Hardcoded security checks

Binder & Ashmem

- Android-specific mechanisms for IPC and shared memory
- Binder
  - Primary IPC mechanism
  - Inspired by BeOS/Palm OpenBinder
- Ashmem
  - Shared memory mechanism
  - Designed to overcome limitations of existing shared memory mechanisms in Linux (debatable)

Android Security Model

- Application-level permissions model
  - Controls access to app components
  - Controls access to system resources
  - Specified by app writers and seen by users
- Kernel-level sandboxing and isolation
  - Isolate apps from each other and the system
  - Prevent bypass of application-level controls
  - Relies on Linux DAC (discretionary access control)
  - Normally invisible to the users and app writers

Discretionary Access Control (DAC)

- Typical form of access control in Linux
- Data-access entirely at the discretion of owner/creator of data
- Some processes (e.g. uid 0) can override and some objects (e.g. sockets) are unchecked
- Based on user & group identity
- Limited granularity, course-grained privilege

Android and DAC

- Restrict use of system facilities by apps
  - e.g. Bluetooth, network, storage access
  - Requires kernel modifications, “special” group IDs
- Isolate apps from each other
  - Unique user and group ID per installed app
  - Assigned to app processes and files
- Hardcoded, scattered policy

Agenda

- Security
- Android Security
- **Case Studies**
- Conclusion

Case study: vold

- vold - Android volume daemon
  - Runs as root
  - Manages mounting of disk volumes
  - Receives netlink messages from the kernel
- CVE-2011-1823
  - Does not verify that message came from kernel
  - Uses signed int from message as array index without checking for < 0
- Demonstrated by the Gingerbreak exploit

GingerBreak: Overview

- Collect information needed for exploitation
  - Identify the vold process
  - Identify addresses and values of interest
- Send carefully-crafted netlink message to vold
  - Trigger execution of exploit binary
  - Create a setuid-root shell
- Execute setuid-root shell
- Got root?? Your @$$ is 0wn3d (technical term)

GingerBreak: Collecting Information

- Identify the vold process
  - /proc/net/netlink to find netlink socket users
  - /proc/pid/cmdline to find vold PID
- Identify addresses and values of interest
  - /system/bin/vold to obtain GOT address range
  - /system/lib/libc.so to find “system” address
  - /etc/vold.fstab to find valid device name
  - logcat to obtain fault address in vold

Case study: ueventd

- ueventd - Android udev equivalent
  - Runs as root
  - Manages /dev directory
  - Receives netlink messages from the kernel
- Same vulnerability as CVE-2009-1185 for udev
  - Does not verify message came from kernel
- Demonstrated by the Exploid exploit

Case study: adbd

- adbd - Android debug bridge daemon
  - Runs as root
  - Provides debug interface
  - Switches to shell UID and executes shell
- Does not check/handle setuid() failure
  - Can lead to a shell running as root
- Demonstrated by RageAgainstTheCage

RageAgainstTheCage: Overview

- Look up adbd process in /proc
- Fork self repeatedly to reach RLIMIT_NPROC for shell identity
- Re-start adbd
- adbd setuid() call fails
- shell runs as root

Case study: zygote

- zygote - Android app spawner
  - Runs as root
  - Receives requests to spawn apps over a socket
  - Uses setuid() to switch to app UID
- Does not check/handle setuid() failure
  - Can lead to app running as root
- Demonstrated by Zimperlich exploit

Zimperlich: overview

- Fork self repeatedly to reach RLIMIT_NPROC for app UID
- Spawn app component via zygote
- Zygote setuid() call fails
- App runs with root UID
  - Re-mounts /system read-write
  - Creates setuid-root shell in /system

Case study: ashmem

- ashmem - anonymous shared memory
  - Android-specific kernel subsystem
  - Used by init to implement shared mapping for system property space
- CVE-2011-1149
  - Does not restrict changes to memory protections
  - Actually two separate vulnerabilities in ashmem
- Demonstrated by KillingInTheNameOf and psneuter exploits

KillingInTheNameOf: Overview

- Change protections of system property space to allow writing
- Modify ro.secure property value
- Re-start adbd
- Root shell via adb

psneuter: Overview

- Set protection mask to 0 (no access) on property space
- Re-start adbd
- adbd cannot read property space
- Defaults to non-secure operation
- Root shell via adb

Agenda

- Security
- Android Security
- Case Studies
- **Conclusion**

Conclusion

- Need more case studies?? I mean, REALLY???
- Android is far from secure: Q.E.D
- So far we're only dealing with the kernel level access controls
- To fully control the apps, need application-layer access controls
- Requires further study of the existing Android security model
- Requires instrumentation of the application frameworks
  - SE Android is a step in right direction
- To protect against system attacks Android should also be bolted to hardware security (e.g. TEE impl.s such as TrustZone)

Thank you!

Q [& possibly] A

Rates chart

- Answers: $1
- Correct answers: $3
- Correct answers requiring thought: $5

References: Android documentation, Stephen S. Smalley, NSA, and various free resources from the Internet, so long as they're free