android Bootcamp 2016

Defense in depth efforts

Nick Kralevich

Thursday January 21, 2016

Agenda

Strategy

Overview of current features

Where we’re going

Why?

Why?

- There will always be bugs.
- Bugs should be hard to exploit.
- Bugs shouldn't be catastrophic.
- Updates are important, but it doesn't solve the whole problem.
- Users and developers expect their data to be safe against all attackers, including compromises of other parts of the system.

Must read: Giant Bags of Mostly Water - Securing your IT Infrastructure by Securing your Team

Four principles of Android Hardening

1. Exploit Mitigation
2. Exploit Containment
3. Attack Surface Reduction
4. Safe-by-default

Exploit Mitigation

Exploit Mitigation

Goals:

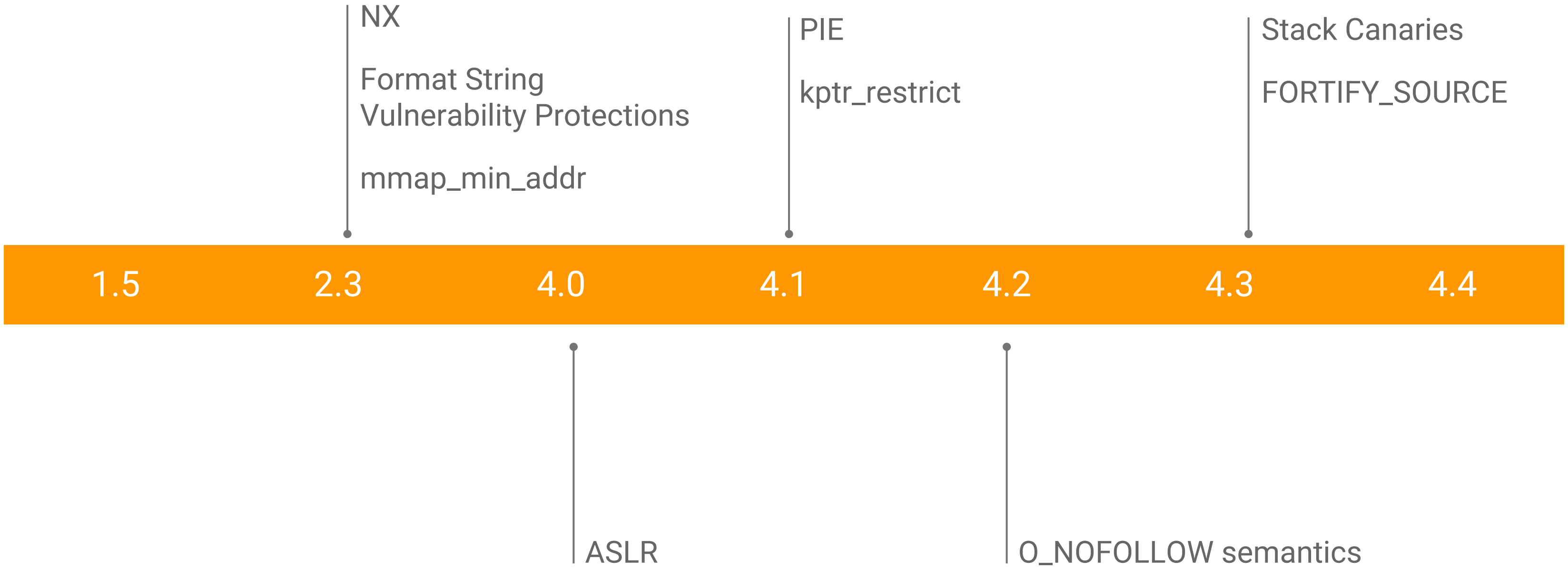
1. Protect against future unknown bugs
2. Make exploits unstable and unreliable
3. Reduce the severity of vulnerabilities
4. Buy additional time to respond

Exploit Mitigation—Historical

- ASLR / PIE (userspace only)
- NX (userspace only)
- Stack Canaries (userspace + kernel)
- FORTIFY_SOURCE (userspace)
- Format String Vulnerability Protections (userspace)
- Mmap_min_addr (kernel)
- RELRO / BIND_NOW (userspace only)
- kptr_restrict (kernel)
- O_NOFOLLOW semantics (userspace)

android

Exploit Mitigation—Historical



Exploit Mitigation—Future

- ASLR improvements (development complete)
  - Greater kernel randomization
  - Link time randomization
- -fsanitize signed/unsigned overflow (enabled for media code)
- -stack-protector-strong (development complete)
- More FORTIFY_SOURCE (ongoing)
- Clang object size detection improvements (compiler work complete)
- CFI (in research)
- Other fsanitize options: bounds, object-size (in research)
- Kernel UDEREF (upstream complete, backport non-trivial)

Exploit Containment

android

Exploit Containment

Goals

- Protect application and user data
- Protect the Android Trusted Computing Base (TCB)

Containment measures

- Limit damage from successful exploitation
- Enforce the principle of least privilege
- Enforce architectural best practices

Exploit Containment—Historical

- UID sandboxes
- SELinux sandboxes
- Privilege separation
- Architectural decomposition

Exploit Containment—Future

- Further SELinux tightening
  - sysfs and debugfs access restrictions
  - More neverallow rules
  - Read access control on property space
- hidepid=2 - limit /proc read access
- seccomp
- mediaserver hardening - privilege decomposition
- execmem removal - no anonymous executable memory
- Hardware back keystore
- Reduction of available ioctl commands

Attack Surface Reduction

android

Attack Surface Reduction

Goals:

1. Make vulnerable code unreachable
2. Buy time for proper fix

Attack Surface Reduction—Historical

- UID sandboxes
- SELinux sandboxes
- Privilege separation
- Architectural decomposition
- Removal of unused functionality
    - Example: Kernel module loading by system_server

Attack Surface Reduction—Future

- SELinux
- Seccomp
- Remove unused kernel functionality
  - System V IPC
- Memory safe language (research)
- Fuzzing

Safe by default

Safe by default

Goals:

1. Make it harder to introduce security bugs
2. Make the easy thing the safe thing

Safe by default—Historical

- File permissions
- Content providers not exported by default
- SELinux
- Verified boot

Safe by default—Future

- Verified Boot enforcing mode and full stack protection
- Data in transit protection: usesCleartextTraffic="false"
- Data at rest protection: disk encryption
- Easier cert pinning/ trust anchor support

THANK YOU