Rooting Android
Lecture 10

Security of Mobile Devices

2018

SMD

SMD

- Linux kernel - DAC security model
- Root user (UID 0) - absolute power
- Root on Android:
    - Bypass Android sandbox
    - Access and modify any file
    - Modify read-only partitions
    - Control system services
    - Remove system applications

SMD

- ▶ Root access not allowed on production devices
- ▶ Limit system processes with root permission
  - ▶ Avoid privilege escalation
- ▶ SELinux - global security policy
  - ▶ Root processes don't have unrestricted access
  - ▶ Still have access to private data
  - ▶ Can modify system behavior
  - ▶ Exploit kernel vulnerability

- Debugging and reverse engineering apps
- System customizations
- Implementing special applications
  - Modifications in the framework or system services
  - Firewall
  - Full device backup
  - Network sharing

SMD

- `ro.build.type` system property
- *user* build:
  - No diagnostics and development tools
  - ADB daemon disabled
  - Debugging only for apps with `debuggable` true
  - No root access using shell
- *userdebug* build:
  - Allows debugging of all apps
  - ADB enabled

SMD

```
marlin:/ $ getprop ro.build.type
user
marlin:/ $ getprop ro.secure
1
marlin:/ $ ps -ef | grep adb
shell         693      1 0 09:42:05 ?        00:00:05 adbd --root_seclabel=u:r:su:s0
marlin:/ $ cat /proc/693/status
Name:    adbd
State:   S (sleeping)
Tgid:    693
Pid:     693
PPid:    1
TracerPid:      0
Uid:     2000    2000    2000    2000
Gid:     2000    2000    2000    2000
Ngid:    0
FDSize:  64
Groups:  1004 1007 1011 1015 1028 3001 3002 3003 3006 3009 3011
[..]
CapInh:  0000000000000000
CapPrm:  0000000000000000
CapEff:  0000000000000000
CapBnd:  00000000000000c0
CapAmb:  0000000000000000
Seccomp:        0
[..]
```

SMD

- Allows debugging of all apps
- ADB enabled
- ro.secure $= 0$
  - ADB daemon continues to run as root
  - Does not drop capabilities - full capability bounding set

SMD

- Obtain root shell
- Run command as another UID
- On *userdebug* without restarting ADB as root
- Default su can be used only by root (0) and shell (2000) users
- Sets UID and GID to 0
- Commands executed from the shell inherit priviledges

SMD

SMD

- In production:
  - *user* build
  - ADB daemon runs as shell user
  - No su command
  - No system/core OS configuration
  - No access to the kernel
- Rooting a device
  - Unlocked bootloader

SMD

1. Write new boot image (custom kernel)
   - *eng* or *userdebug*
   - `ro.secure`, `ro.debuggable`
   - Current *user* builds disable ADB root
   - ADB daemon ignores properties
2. Unpack system image, write su, write system partition
   - Would allow su access from third-party apps
   - From Android 4.3, system mounted with `nosetuid`
   - Apps cannot execute SUID programs
   - Zygote processes without capabilities

- SELinux enforcing mode:
  - Execute with root -> security context unchanged, MAC policy
- SELinux must be disabled for root access
- Disable security measures
- No system updates from the manufacturer

SMD

- OTA package - add and modify system files
- Without rewriting the whole OS
- Superuser apps:
  - OTA package to be installed
  - Manager application (updatable)

SMD

- ▶ OTA package and application
- ▶ Developed by Jorrit "Chainfire" Jongma
- ▶ Actively maintained
- ▶ Most popular

- Native binaries for ARM and x86
- Scripts for installing and starting SuperSU daemon
- Apk of management application
- Updater scripts
  - `update-binary, updater-script`

- Mount rootfs, system and data partitions as read-write
- Copy `su` and `daemonsu` to `/system/xbin/`
- Copy apk to `/system/app/` -> installed at reboot
- Copy `install-recovery.sh` to `/system/etc/`

- Set permissions and SELinux security labels of installed binaries
  - `u:object_r:system_file:s0 label`
- Call `/system/xbin/su --install`
  - Post-install initialization
- Unmount system and data partitions

Samsung Galaxy S7 Edge

```
hero2lte:/ # cat init.supersu.rc
# earliest possible SuperSU daemon launch, with fallback to service
on post-fs-data
    exec u:r:supersu:s0 root root -- /sbin/launch_daemonsu.sh post-fs-data

# mount /data/su.img to /su
on property:sukernel.mount=1
    mount ext4 loop@/data/su.img /su noatime

# launch SuperSU daemon
service daemonsu /sbin/launch_daemonsu.sh service
    class late_start
    user root
    seclabel u:r:supersu:s0
    oneshot
```

SMD

- Bypass security constraints (no capabilities, SELinux)
- Usage:
  - Apps use su binary
  - Sends commands through socket to daemonsu
  - Executes commands as root in u:r:supersu:s0 SELinux context

SMD

Samsung Galaxy S7 Edge

```
hero2lte:/ $ ps -Z
LABEL    USER   PID    PPID   VSIZE   RSS   WCHAN   PC   NAME
u:r:supersu:s0     root      3217   1      7664    660
__skb_recv 0000000000 S daemonsu:mount:master
u:r:supersu:s0     root      3224   1      11776   672
__skb_recv 0000000000 S daemonsu:master
[..]
u:r:untrusted_app:s0:c512,c768 u0_a192    6687   3281
1848480 91888 SyS_epoll_ 0000000000 S eu.chainfire.supersu
[..]
u:r:shell:s0       shell     24047 19245 7664    1936   poll_sched 7c09943a78 S su
u:r:supersu:s0     root      24050 19251 16032   1816
poll_sched 77574aea78 S daemonsu:0:24047
u:r:supersu:s0     root      24051 24050 7736    1752   poll_sched 78d326ca78 S sush
_____

hero2lte:/ $ su -c sleep 100
```

SMD

- ▶ Run `su -c sleep 100` from shell
- ▶ su process executes as shell, in the `u:r:shell:s0` domain
- ▶ su sends command to `daemonsu:0:24047`
  - ▶ An instance of `daemonsu` created to run the command
- ▶ `daemonsu` executes as root, in `u:r:supersu:s0` domain
- ▶ `daemonsu` executes starts sush to run sleep
  - ▶ sush executes as root, in `u:r:supersu:s0` domain

- `eu.chainfire.supersu` process
- Asks the user to grant root access when needed
- One time, a period, permanently
- White list of applications

- CyanogenMod (now LineageOS)
- Start su as daemon in `init.superuser.rc`
  - `service su_daemon /system/xbin/su --daemon`
- Service started or stopped using `persist.sys.root_access` property
  - Value 0-3 - root access to none, apps, adb shell, or both

SMD

SMD

- Settings -> Enable Developer mode, OEM unlocking, USB debugging
- Download TWRP for hero2lte (code name)
- Enter Download/Odin mode by pressing [Volume Down] + [Home] + [Power]
- Write TWRP on device using Odin (Windows)
- Immediately enter Recovery mode by pressing [Volume Up] + [Home] + [Power]

SMD

- In TWRP - format data partition
- Push SuperSU and no-verity archives to sdcard
- Install no-verity using TWRP
- Install SuperSU using TWRP
- Reboot - it might take a little while

- Download the latest SuperSU application
- Download the latest TWRP application
- From SuperSU, grant permissions to apps that require root
- `adb shell -> $ su -> #`

SMD

▶ Android Security Internals, Nicolay Elenkov, 2015

▶ Android Hacker's Handbook, Joshua J. Drake, 2014

SMD

- Root access
- SELinux
- User build
- Engineering build

- SuperSU
- TWRP
- LineageOS