



Defeating Security Enhancements (SE) for Android

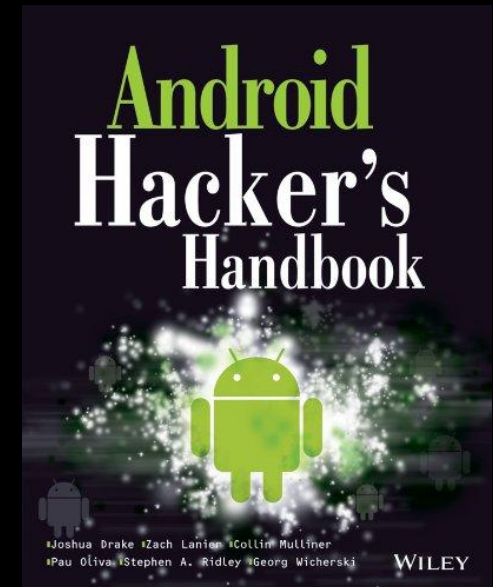
Pau Oliva Fora

poliva@viaforensics.com

[@pof](#)

\$ whoami

- Pau Oliva Fora, aka @pof
 - Mobile security engineer with viaForensics
 - Linux guy, R+D background
 - Smartphone research since 2004
 - Android research since 2008
 - Co-author of the upcoming Android Hacker's Handbook



AGENDA

1. Tested implementations
2. Effectiveness
3. Weaknesses
4. Implementation issues

1. TESTED IMPLEMENTATIONS

TESTED IMPLEMENTATIONS

Compiled from public Sources
(AOSP + bitbucket)

TESTED IMPLEMENTATIONS

SEAdmin vs. SEManager

TESTED IMPLEMENTATIONS

Toshiba AT300
(sealime: Linux Security Module)

2. EFFECTIVENESS

EFFECTIVENESS

Good to enforce fine-grained
Mandatory Access Control (MAC)

- Install time MAC
- Intent MAC
- Content Provider MAC

EFFECTIVENESS

Prevent privilege escalations by
isolating “contexts”

EFFECTIVENESS

Permission checks on IPC operations
(binder)

EFFECTIVENESS

Permission revocation

3. WEAKNESSES

WEAKNESSES

Known:

Doesn't protect against kernel vulns

WEAKNESSES

Needs to be enhanced:

Secure Boot + runtime integrity check

WEAKNESSES

Multiple workarounds in
commercial implementations:

Vendors don't know how to write policies

4. IMPLEMENTATION ISSUES

IMPLEMENTATION ISSUES

Issue #1: don't forget recovery!

System boots in Enforcing mode

BUT...

IMPLEMENTATION ISSUES

Issue #1: don't forget recovery!

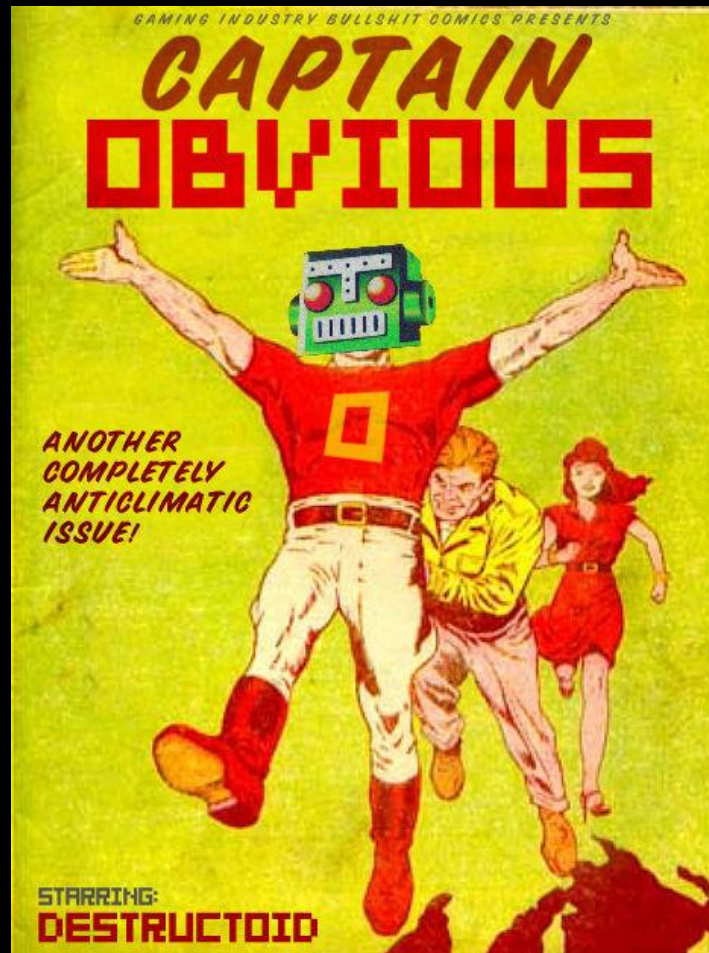
System boots in Enforcing mode

BUT...

...recovery image left in Permissive mode

IMPLEMENTATION ISSUES

Issue #1: don't forget recovery!



DEMO 0:
so obvious
it doesn't
need
a demo!

IMPLEMENTATION ISSUES

Issue #2: check your policies!

root user can't disable SEAndroid

BUT...

IMPLEMENTATION ISSUES

Issue #2: check your policies!

root user can't disable SEAndroid

BUT...

...system user CAN

IMPLEMENTATION ISSUES

Issue #2: check your policies!

root user can't disable SEAndroid

BUT...

...system user CAN *facepalm*

IMPLEMENTATION ISSUES

Issue #2: check your policies!

DEMO 1:

Disable SELinux with root or system privileges



IMPLEMENTATION ISSUES

Issue #2: check your policies!

DEMO 1:

Disable SELinux with root or system privileges

```
root@android# id
uid=0 gid=0 (root)
root@android# setenforce 0
Permission denied
root@android# echo 0 > /sys/fs/selinux/enforce
Permission denied
root@android# su system
system@android$ setenforce 0
```

IMPLEMENTATION ISSUES

Issue #3: never enforce from a system app!

SEAndroid enforced from a system app

IMPLEMENTATION ISSUES

Issue #3: never enforce from a system app!

SEAndroid enforced from a system app



DOUBLE FACEPALM

When the Fail is so strong, one Facepalm is not enough.

IMPLEMENTATION ISSUES

Issue #3: never enforce from a system app!

Combine it with fail #1 and...

```
$ adb reboot recovery
```

```
$ adb wait-for device
```

```
$ adb pull /system/app/SEAndroidManager.apk
```

```
$ adb remount
```

```
$ adb shell rm /system/app/SEAndroidManager.apk
```

```
$ adb reboot
```

IMPLEMENTATION ISSUES

Issue #3: never enforce from a system app!

Combine it with fail #1 and...

```
$ adb reboot recovery  
$ adb wait-for device  
$ adb pull /system/app/SEAndroidManager.apk  
$ adb remount  
$ adb shell rm /system/app/SEAndroidManager.apk  
$ adb reboot
```

BUT what if we don't have access to recovery?

IMPLEMENTATION ISSUES

Issue #3: never enforce from a system app!

DEMO 2:

Wrong chain of trust → disable during boot

IMPLEMENTATION ISSUES

Issue #3: never enforce from a system app!

Not protected from being disabled during boot:

```
$ adb reboot  
$ while true; do  
adb shell pm disable com.android.seandroid_manager ;  
done
```

You can even over-complicate that and write an android app with a higher priority boot receiver...

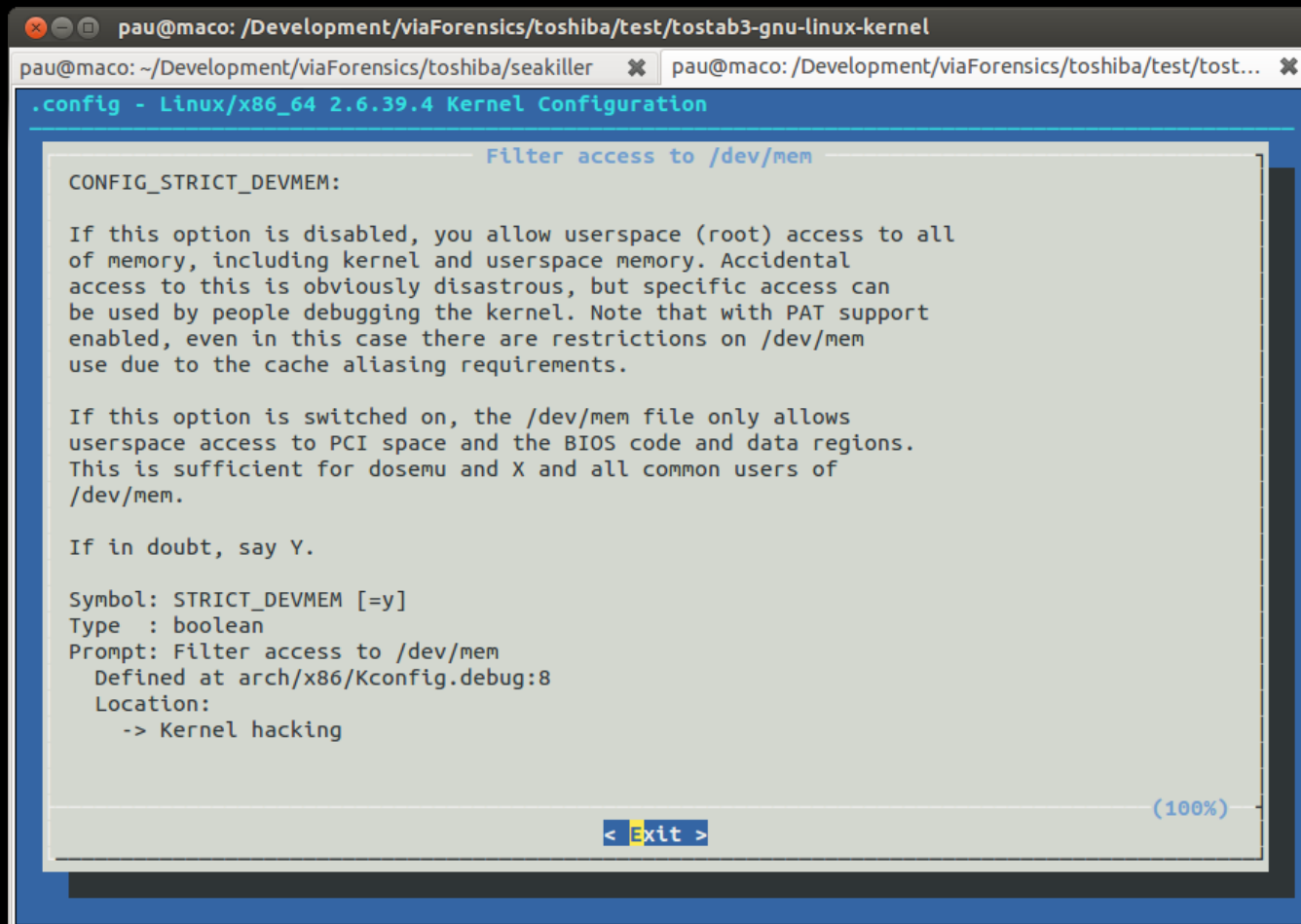
IMPLEMENTATION ISSUES

Issue #4: Toshiba FAIL

```
$ adb shell  
root@android# ls -l /proc/seandroid  
opendir failed, Operation not permitted
```


IMPLEMENTATION ISSUES

Issue #4: Toshiba FAIL



```
pau@maco: /Development/viaForensics/toshiba/test/tostab3-gnu-linux-kernel
pau@maco: ~/Development/viaForensics/toshiba/seakiller x pau@maco: /Development/viaForensics/toshiba/test/tost... x
.config - Linux/x86_64 2.6.39.4 Kernel Configuration
Filter access to /dev/mem
CONFIG_STRICT_DEVMEM:
If this option is disabled, you allow userspace (root) access to all
of memory, including kernel and userspace memory. Accidental
access to this is obviously disastrous, but specific access can
be used by people debugging the kernel. Note that with PAT support
enabled, even in this case there are restrictions on /dev/mem
use due to the cache aliasing requirements.
If this option is switched on, the /dev/mem file only allows
userspace access to PCI space and the BIOS code and data regions.
This is sufficient for dosemu and X and all common users of
/dev/mem.
If in doubt, say Y.
Symbol: STRICT_DEVMEM [=y]
Type : boolean
Prompt: Filter access to /dev/mem
Defined at arch/x86/Kconfig.debug:8
Location:
-> Kernel hacking
(100%)
< Exit >
```

IMPLEMENTATION ISSUES

Issue #4: Toshiba FAIL

DEMO 3:

Disable SEAndroid LSM
by poking kernel memory

Thank you!

Contact: [@pof](#)

poliva@viaforensics.com

Greetz & thanks: @djrbliss, @timstrazz, @TeamAndIRC, @cryptax, @ChainfireXDA, @jduck, @quine, @collinrm, @ochsff, @s7ephen, @iolandatweets, @thomas_cannon, @insitusec, @marcograss, @ahoog42, @0xroot, @andreybelenko, @giantpune & vF team!



VIAFORENSICS

advancing mobile security