Thwarting unknown bugs: hardening features in the mainline Linux kernel

ARM

Mark Rutland <mark.rutland@arm.com>
ARM Ltd

Embedded Linux Conference Europe 2016

October 3, 2016

What's the problem?

Linux has bugs today

```
git log --oneline \
--grep='Fixes:'    \
v4.7..v4.8-rc1 |   \
wc -l

503
```

Some bugs have security implications

There are **1496** CVE entries that match your search.

Name	Description
CVE-2016-6516	Race condition in the ioctl_file_dedupe_range function in fs/ioctl.c in the Linux kernel through 4.7 allows local users to cause a denial of service (heap-based buffer overflow) or possibly gain privileges by changing a certain count value, aka a "double fetch" vulnerability.
CVE-2016-6480	Race condition in the ioctl_send_fib function in drivers/scsi/aacraid/commctrl.c in the Linux kernel through 4.7 allows local users to cause a denial of service (out-of-bounds access or system crash) by changing a certain size value, aka a "double fetch" vulnerability.
CVE-2016-6198	The filesystem layer in the Linux kernel before 4.5.5 proceeds with post-rename operations after an OverlayFS file is renamed to a self-hardlink, which allows local users to cause a denial of service (system crash) via a rename system call, related to fs/namei.c and fs/open.c.
CVE-2016-6197	fs/overlayfs/dir.c in the OverlayFS filesystem implementation in the Linux kernel before 4.6 does not properly verify the upper dentry before proceeding with unlink and rename system-call processing, which allows local users to cause a denial of service (system crash) via a rename system call that specifies a self-hardlink.
CVE-2016-6187	The apparmor_setprocattr function in security/apparmor/lsm.c in the Linux kernel before 4.6.5 does not validate the buffer size, which allows local users to gain privileges by triggering an AppArmor setprocattr hook.
CVE-2016-6162	net/core/skbuff.c in the Linux kernel 4.7-rc6 allows local users to cause a denial of service (panic) or possibly have unspecified other impact via certain IPv6 socket operations.
CVE-2016-6156	Race condition in the ec_device_ioctl_xcmd function in drivers/platform/chrome/cros_ec_dev.c in the Linux kernel before 4.7 allows local users to cause a denial of service (out-of-bounds array access) by changing a certain size value, aka a "double fetch" vulnerability.
CVE-2016-6136	Race condition in the audit_log_single_execve_arg function in kernel/auditsc.c in the Linux kernel through 4.7 allows local users to bypass intended character-set restrictions or disrupt system-call auditing by changing a certain string, aka a "double fetch" vulnerability.
CVE-2016-6130	Race condition in the sclp_ctl_ioctl_sccb function in drivers/s390/char/sclp_ctl.c in the Linux kernel before 4.6 allows local users to obtain sensitive information from kernel memory by changing a certain length value, aka a "double fetch" vulnerability.
CVE-2016-5829	Multiple heap-based buffer overflows in the hiddev_ioctl_usage function in drivers/hid/usbhid/hiddev.c in the Linux kernel through 4.6.3 allow local users to cause a denial of service or possibly have unspecified other impact via a crafted (1) HIDIOCGUSAGES or (2) HIDIOCSUSAGES ioctl call.
CVE-2016-5828	The start_thread function in arch/powerpc/kernel/process.c in the Linux kernel through 4.6.3 on powerpc platforms mishandles transactional state, which allows local users to cause a denial of service (invalid process state or TM Bad Thing exception, and system crash) or possibly have unspecified other impact by starting and suspending a transaction before an exec system call.
CVE-2016-5728	Race condition in the vop_ioctl function in drivers/misc/mic/vop/vop_vringh.c in the MIC VOP driver in the Linux kernel before 4.6.1 allows local users to obtain sensitive information from kernel memory or cause a denial of service (memory corruption and system crash) by changing a certain header, aka a "double fetch" vulnerability.

(``linux kernel" CVEs on mitre.org, 2016-09-27 - `https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=linux+kernel`)

Adversaries find bugs before we do

[redacted] vs x86_64 Linux Kernel

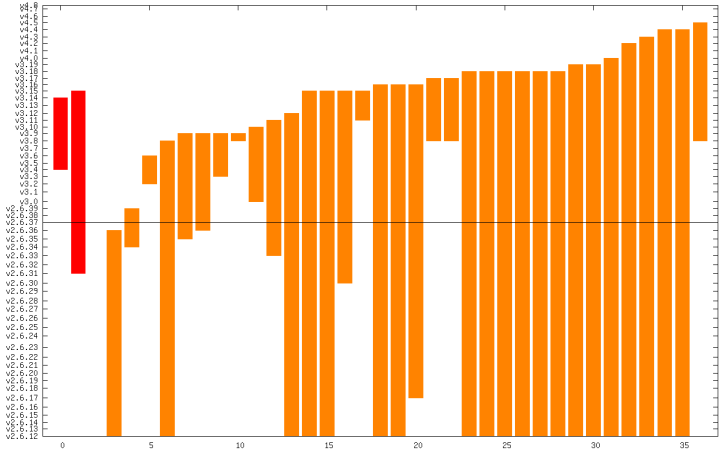
From: [redacted]
Date: Wed, 15 Sep 2010 22:08:23 -0700 (PDT)

```
/*

[redacted] Vs Linux Kernel x86_64 0day

Today is a sad day..

R.I.P.
Tue, 29 Apr 2008  /  Tue, 7 Sep 2010

a bit of history:
MCAST_MSFILTER Compat mode bug found... upon commit! (2 year life on this one)
---------------------------------------------------------------

Thanks you for signing-off on this one guys.

This exploit has been tested very thoroughly
over the course of the past few years on many many targets.

Thanks to redhat for being nice enough to backport it into early
kernel versions (anything from later August 2008+)

Exploit attached. Another 0day bites the dust and goes into our public exploit pack :)
[redacted] brings you ABftw.c - Linux Kernel x86_64 local not0dayanymore exploit.
```

Attachment: ABftw.c
Description:

(Trimmed and redacted announcement - http://seclists.org/fulldisclosure/2010/Sep/268)

Bugs go unnoticed upstream for a long time



(Kees Cook, LSS2016, `Status of the Kernel Self Protection Project' - `https://outflux.net/slides/2016/lss/kspp.pdf`)

The presence of bugs is unavoidable

- Code written by experienced engineers has bugs
- Code reviewed by subject-matter experts has bugs
- Static analysis only finds some bugs
- Testing and fuzzing only finds some bugs
- Formal methods do not scale to size and scope of project
  (30+ architectures with varied ISAs, memory models, system-level details)

All are valuable, but insufficient to rule out bugs entirely.

The big picture

- Linux is being attacked in the wild
- ... via bugs we don't know about (yet)
- ... which we can't hope to avoid (yet)

Products (and their users) can be vulnerable for their entire lifetime

Hardening

Making unknown bugs harder to exploit

- Target common classes of error (e.g. accidental `__user` pointer dereference)
- Have common code prevent and/or detect this
- When detected, prevent further badness somehow (e.g. `panic()`)

- Reduces exploitability, but doesn't fix underlying bugs
- Complementary to usual bug hunting

Hardening in mainline

- Some hardening features have made it to mainline
  - Supported upstream
  - Lower maintenance burden
- ... but many are not enabled by default (yet)
  - You could be missing out on free protection today
  - Easy to be put off by unclear tradeoffs and precevied issues
- ... and only "recently"
  - Most phones aren't running v4.8 yet...
  - Easy to miss if you're not paying attention

The rest: coming soon

ARM