Using Static Checking To Find Security Vulnerabilities In The Linux Kernel

Linuxcon Europe 2016

Vaishali Thakkar

(vaishali.thakkar@oracle.com)

Self Introduction

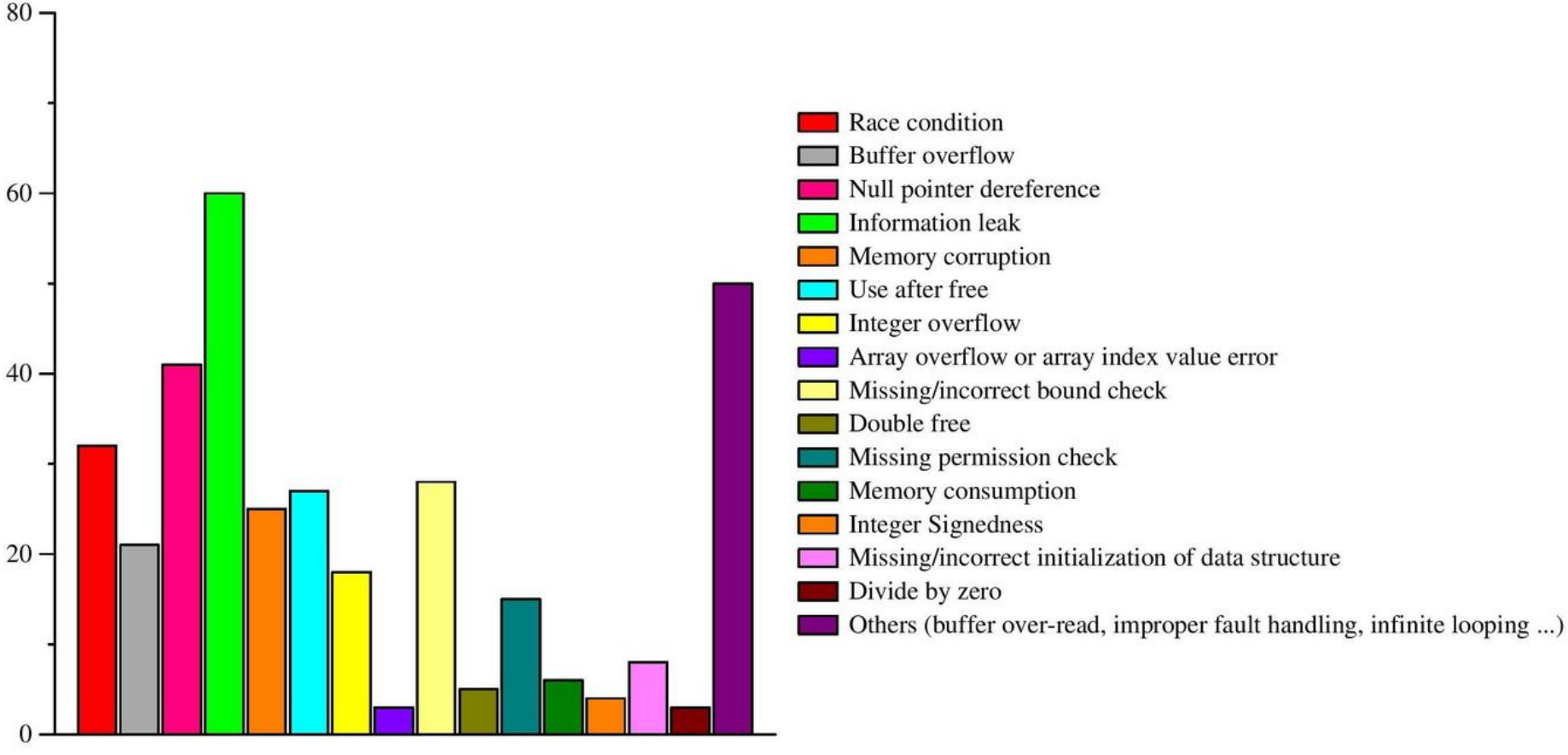
- Linux Kernel developer at Oracle
- Working in kernel security engineering group and memory management
- Interested in many different subsystems of the Linux Kernel

Agenda

- Overview of security issues in the Linux Kernel
- Static checking
- Static checking tools
- Automated checking
- Bonus

Cause of the kernel bugs

Data: Jan, 2014 to August, 2016 [cvedetails.com]



Language-specific security issues

- Buffer overflow [stack and heap based]
- Use after free and double free
- Null pointer dereference and invalid pointer dereference
- String issues
  - Incorrect/missing bound check, array overflow, out-of-bound errors etc
- Others
  - Integer signedness, buffer over read, deadlock, array index value error etc

General security issues

- Race conditions
- Memory corruption and memory consumption
- Divide by zero and off by one
- Integer overflow
- Information leak

Linux kernel specific security issues

- Incorrect/missing initialization of data structure
- Calling sleeping functions under invalid context
- Missing permission check
- Uninitialized data
- Others
  - Infinite looping, improper fault handling, copy pasted code, etc

Static code checking

Static code analysis

- Usually performed as part of a code review and is carried out at the implementation phase of a security development lifecycle (SDL).
- Performed without actually executing programs.
- Benefits: Find bugs early, cheaper to fix the bugs when they are caught at the early stage of software development
- Things to care about: False positives

Why static checkers?

- Example one:

```
diff --git a/security/keys/key.c b/security/keys/key.c
index bd5a272..346fbf2 100644
--- a/security/keys/key.c
+++ b/security/keys/key.c
@@ -597,7 +597,7 @@ int key_reject_and_link(struct key *key,

        mutex_unlock(&key_construction_mutex);

-       if (keyring)
+       if (keyring && link_ret == 0)
                __key_link_end(keyring, &key->index_key, edit);

        /* wake up anyone waiting for a key to be constructed */
```

Commit 38327424b40bce by Dan Carpenter, reported by Smatch. Fixes CVE-2016-4470

Why static checkers?

- Example one:

```
int key_reject_and_link(...)
...
if (keyring) {
                if (keyring->restrict_link)
                        return -EPERM;

                link_ret = __key_link_begin(keyring,
                                            &key->index_key, &edit);
        }
...
if (keyring && link_ret == 0)
                __key_link_end(keyring, &key->index_key, edit);
```

Missing check? Potential uninitialized variable? What is so special about this?

Why static checkers?

- Example one: security/keys/keyring.c

```
int __key_link_begin(..., ... , struct assoc_array_edit **_edit)
...
{
      struct assoc_array_edit *edit;
      ...
      edit = assoc_array_insert(&keyring->keys,
                                &keyring_assoc_array_ops,
                                index_key, NULL);
      ...
      if (!edit->dead_leaf) {
      ret = key_payload_reserve(keyring,
      keyring->datalen + KEYQUOTA_LINK_BYTES);
      if (ret < 0)
      goto error_cancel;
      ...
error_cancel:
      assoc_array_cancel_edit(edit);
```

Failure of __key_link_begin = uninitialization of 'edit' = system crash by local users

Why static checkers?

- Example two:

```
--- a/drivers/net/wireless/realtek/rtlwifi/rtl8188ee/dm.c
+++ b/drivers/net/wireless/realtek/rtlwifi/rtl8188ee/dm.c
@@ -1790,6 +1790,7 @@void rtl88e_dm_watchdog(...)
        if (ppsc->p2p_ps_info.p2p_ps_mode)
                fw_ps_awake = false;

+       spin_lock(&rtlpriv->locks.rf_ps_lock);
        if ((ppsc->rfpwr_state == ERFON) &&
            ((!fw_current_inpsmode) && fw_ps_awake) &&
            (!ppsc->rfchange_inprogress)) {
@@ -1802,4 +1803,5 @@void rtl88e_dm_watchdog(...)
                rtl88e_dm_check_edca_turbo(hw);
                rtl88e_dm_antenna_diversity(hw);
        }
+       spin_unlock(&rtlpriv->locks.rf_ps_lock);
 }
```

Commit 204e2ab22e1e2d0 by Larry Finger, reported by LDV tools

Why static checkers?

- Example two: drivers/net/wireless/rtlwifi/rtl8188ee/hw.c

```
bool rtl88ee_gpio_radio_on_off_checking(...)
{
...
spin_lock(&rtlpriv->locks.rf_ps_lock);
       if (ppsc->rfchange_inprogress) {
            spin_unlock(&rtlpriv->locks.rf_ps_lock);
            return false;
        } else {
```

Potential race condition

Why static checkers?

- Example two: drivers/net/wireless/rtlwifi/rtl8188ee/hw.c

```
bool rtl88ee_gpio_radio_on_off_checking(...)
{
...
spin_lock(&rtlpriv->locks.rf_ps_lock);
       if (ppsc->rfchange_inprogress) {
            spin_unlock(&rtlpriv->locks.rf_ps_lock);
            return false;
        } else {
```

Similar code was present in 5 other files

Static checking tools

scripts/checkpatch.pl

- Written by Andy Whitcroft, Joe Perches
- Checks for basic coding style issues and sometimes for incorrect API usuage
- Warns about a few errors that can trigger security bugs:
  - Misuse of memsets, check for lockdep_set_novalidate_class, Prefixing 0x with decimal output, using weak declarations which can have unintended link defects
- Good to run it for new submissions

scripts/checkpatch.pl

- Example output: scripts/checkpatch.pl --file --terse <path_to_directory>

```
drivers/staging/media/bcm2048/radio-bcm2048.c:307: ERROR: Use 4
digit octal (0777) not decimal permissions

drivers/staging/media/bcm2048/radio-bcm2048.c:1539: CHECK: Avoid
crashing the kernel - try using WARN_ON & recovery code rather
than BUG() or BUG_ON()

drivers/staging/media/bcm2048/radio-bcm2048.c:1997: ERROR: Macros
with complex values should be enclosed in parentheses

drivers/staging/media/bcm2048/radio-bcm2048.c:2025: WARNING:
Prefer 'unsigned int' to bare use of 'unsigned'

drivers/staging/media/bcm2048/radio-bcm2048.c:2543: WARNING:
struct v4l2_ioctl_ops should normally be const
```

Sparse

- Written by Linus Torvalds, later maintained by Josh Triplett, Chris Li
- Provides a set of annotations designed to convey semantic information about types.
  - For example, what address space pointers point to or what locks a function acquires or releases.
- More than 6000 patches accepted so far.
- Documentation: https://kernelnewbies.org/Sparse

Sparse

- Can find the following security or related bugs:
  - Warns about casts that add an address space to a pointer type and truncate const values
  - Warns about unsupported operations or type mismatches with restricted integer types.
  - Warns about any non-static variable or function definition that has no previous declaration.
  - Warns about the use of 0 as a NULL pointer.

Sparse

- Example output: make C=2 <path_to_directory>

```
drivers/staging/wlan-ng/p80211conv.c:132:25: warning: cast to
restricted __be16

drivers/staging/wlan-ng/p80211conv.c:154:38: warning: incorrect
type in assignment (different base types)
drivers/staging/wlan-ng/p80211conv.c:154:38: expected unsigned
short [unsigned] [usertype] type
drivers/staging/wlan-ng/p80211conv.c:154:38: got restricted
__be16 [usertype] <noident>

drivers/staging/wlan-ng/prism2fw.c:251:15: warning: memset with
byte count of 120000

drivers/staging/lustre/lnet/selftest/rpc.c:764:9: warning: context
imbalance in 'srpc_shutdown_service' - different lock contexts for
basic block
```

Smatch

- Written by Dan Carpenter
- More than 3000 bugs fixed by Smatch, mostly by Dan
- Uses sparse as a C parser
- Documentation:

  https://blogs.oracle.com/linuxkernel/entry/smatch_static_analysis_tool_overview

Smatch

- Can find the following security or related bugs:
  - Null pointer dereference, error pointer dereference, buffer overflow etc
  - Off by one bugs
  - Locking related bugs - Double locks/unlocks, missing unlock etc
  - Unintialized variable/data and signedness related bugs
  - Use after free, double free etc
  - Information leak
  - Unnecessary null check and missing null check

Smatch

- Example output: <path_to_smatch>/smatch_scripts/kchecker --spammy ./

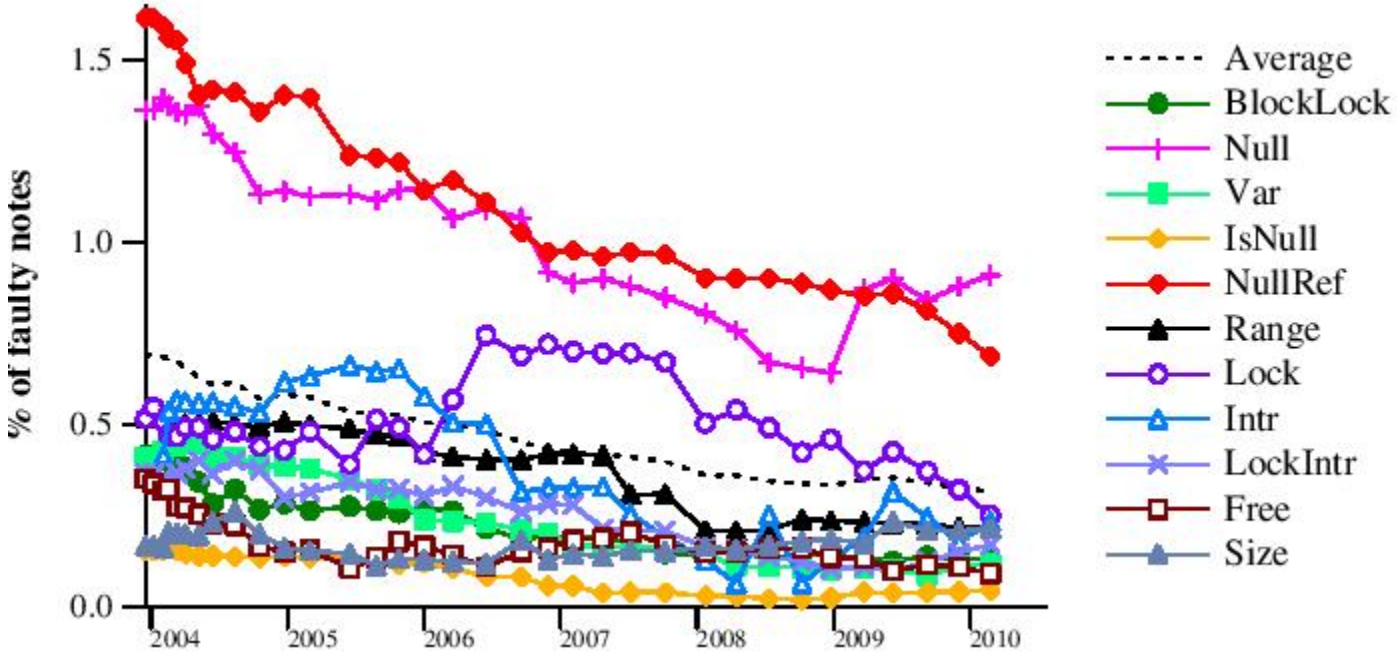
```
drivers/staging/xgifb/vb_setmode.c:3581 XGI_SetGroup2() warn: mask
and shift to zero

drivers/staging/xgifb/vb_setmode.c:5334 XGI_EnableBridge() warn:
we tested 'pVBInfo->VBInfo & 256' before and it was 'true'

drivers/staging/vt6656/rf.c:876 vnt_rf_table_download() error:
memcpy() 'addr1' too small (3 vs 48)

drivers/staging/rts5208/ms.c:2736 ms_build_l2p_tbl() error:
buffer overflow 'ms_start_idx' 17 <= s32max

drivers/staging/rts5208/ms.c:2594 ms_build_l2p_tbl() error: we
previously assumed 'ms_card->segment' could be null(see line 2586)

drivers/staging/rts5208/sd.c:4115 ext_sd_send_cmd_get_rsp() warn:
masked condition '(*ptr + 3 & 30) != 3' is always true.
```

Coccinelle

- Written by Julia Lawall
- Pattern matching and transformation tool
- Can warn you about bugs [report mode] or suggest a fix for the bugs [patch mode]
- More than 4000 patches fixed by Coccinelle
- Documentation: http://coccinelle.lip6.fr/

Coccinelle

Some of the fault types found by Coccinelle



Coccinelle

- Can find the following security or related bugs:
  - Null pointer dereference
  - Use after free
  - Locking related bugs - Double locks/unlocks, missing unlock etc
  - Use of sleeping functions or GFP_KERNEL flag under the lock
  - Use after free, double free etc
  - Protecting function pointers in data structures

Coccinelle

- Example output: make coccicheck <path_to_directory>

```
./security/integrity/ima/ima_template.c:192:29-35: ERROR:
application of sizeof to pointer

./drivers/power/supply/ab8500_charger.c:3676:8-28: ERROR:
Threaded IRQ with no primary handler requested without
IRQF_ONESHOT

./sound/soc/samsung/i2s.c:1269:2-4: ERROR: test of a variable
/field address

./drivers/block/loop.c:736:8-15: ERROR: PTR_ERR applied after
initialization to constant on line 728

./fs/btrfs/send.c:6335:22-39: ERROR: sctx is NULL but
dereferenced.

./drivers/misc/lkdtm_heap.c:38:1-5: ERROR: reference preceded
by free on line 37
```

GCC6

- Some new useful warnings
- Warns about a few errors which can trigger security[1] bugs:
  - Null pointer dereference[-Wnull-dereference], left shift of the negative value[-Wshift-negative-value], left shift overflow[-Wshift-negative-value] etc.
- Documentation: https://gnu.wildebeest.org/blog/mjw/2016/02/15/looking-forward-to-gcc6-many-new-warnings/

LDV[Linux driver verification] tools

- The LDV tools static verification framework analyzes Linux kernel modules and detects errors.
- Project by Russian Linux Verification Center, supported by Linux Foundation. Based at the Institute for System Programming of the Russian Academy of Sciences (ISPRAS)
- Around 240 patches accepted into the Linux Kernel
- Documentation: http://linuxtesting.org/results/ldv

LDV [Linux driver verification] tools

- Can find the following security or related bugs:
  - Race conditions
  - Memory leaks and resource leaks
  - Locking related bugs - Double locks/unlocks, missing unlock etc
  - Use of sleeping functions in the atomic context and deadlocks
  - Null pointer dereference
  - Uninitialized variables

Automatic checking

0 day testing robot

- Maintained by Fengguang Wu
- Tests patch submissions in the mailing lists
- Covers many aspects of the Linux kernel
- For the monitored git trees, 0-Day reports build failures, boot failures, functional bugs, and regression/improvement of kernel performance.

0 day testing robot

- Notifies patch author with failure information and steps to reproduce the failure
- Runs some coccinelle scripts as well
- Sometime sends patches too

0 day testing robot

- Example report output:

```
From: kbuild test robot <lkp@intel.com>
Re: [PATCH V5 2/2] thermal: max77620: Add thermal driver for
reporting junction temp
Hi Laxman,

[auto build test WARNING on thermal/next]
[also build test WARNING on next-20160823]
[cannot apply to v4.8-rc3]
[if your patch is applied to the wrong git tree, please drop
us a note to help improve the system]
[Suggest to use git(>=2.9.0) format-patch --base=<commit>
(or --base=auto for convenience) to record what (public,
well-known) commit your patch series was built on]
[Check https://git-scm.com/docs/git-format-patch for more
information]
```

To be continued..

0 day testing robot

- Example report output:

```
rl:    https://github.com/0day-ci/linux/commits/Laxman-Dewangan/
thermal-max77620-Add-DT-binding-doc-for-thermal-driver/
20160823-151342
base:   https://git.kernel.org/pub/scm/linux/kernel/git/rzhang/
linux.git next
config: x86_64-allmodconfig (attached as .config)
compiler: gcc-6 (Debian 6.1.1-9) 6.1.1 20160705
reproduce:
        # save the attached .config to linux build tree
        make ARCH=x86_64

All warnings (new ones prefixed by >>):

   drivers/thermal/max77620_thermal.c: In function
'max77620_thermal_probe':
>> drivers/thermal/max77620_thermal.c:95:5: warning: 'mtherm'
is used uninitialized in this function [-Wuninitialized]
     if (!mtherm)
        ^
```

0 day testing robot

- Example automated patch output: commit e014e846855223

```
Author: Wu Fengguang <fengguang.wu@intel.com>
Date: Sat Mar 19 00:54:50 2016 +0800
 ovs: internal_set_rx_headroom() can be static

 Signed-off-by: Fengguang Wu <fengguang.wu@intel.com>
 Signed-off-by: David S. Miller <davem@davemloft.net>
--- a/net/openvswitch/vport-internal_dev.c
+++ b/net/openvswitch/vport-internal_dev.c
@@ -138,7 +138,7 @@ internal_get_stats(struct net_device
*dev, struct rtnl_link_stats64 *stats)
 return stats;
 }
-void internal_set_rx_headroom(struct net_device *dev,
int new_hr)
+static void internal_set_rx_headroom(struct net_device
*dev, int new_hr)
 {
 dev->needed_headroom = new_hr;
 }
```

Bonus: Fuzzers

Trinity

- Developed by Dave Jones
- Creates a list of file descriptors instead of passing it as an argument. And when a syscall needs an fd, it will pass one of fd randomly.
- Also shares those file descriptors between multiple processes.
- File descripters are not only thing it knows about, every syscall had arguments annotated

Trinity

- Capable of finding the following security or related bugs:
  - OOPS

    [ex. CVE-2010-4256, c66fb347946ebdd5b10908866ecc9fa05ee2cf3d]
  - Locking related bugs like broken locking, recursive locking etc.
  - Error path memory leaks
  - Hardware bugs

Syzkaller

- Developed by Dmitry Vyukov and a team [Google]
- Unsupervised, coverage-guided Linux syscall fuzzer. Meant to be used with KASAN.
- More than 200 bugs fixed so far
- Documentation: https://github.com/google/syzkaller

Syzkaller

- Can find the following security or related bugs:
  - Deadlocks
  - Sleeping functions called from invalid context or under the atomic context
  - Infinite looping
  - Use after free
  - Resource leaks, memory leaks and information leaks
  - Null dereference

Some other fuzzers/sanitizers

- ThreadSanitizer: For detecting data races
- AFL: Succesful for user space code, can be used on the kernel side[1]
- Address sanitizer: For detecting memory access bugs

[1]https://lwn.net/Articles/685182/

Conclusion

- **Smatch:** https://blogs.oracle.com/linuxkernel/entry/smatch_static_analysis_tool_
- **Sparse:** https://kernelnewbies.org/Sparse
- **Coccinelle:** http://coccinelle.lip6.fr/, Documentation/coccinelle.txt
- **GCC6:** GNU Blog
- **LDV Tools:** http://linuxtesting.org/ldv
- **Trinity:** http://codemonkey.org.uk/projects/trinity/
- **Syzkaller:** https://github.com/google/syzkaller

Questions?

Thank You