serverfault

Kernel stack trace to source code lines

Asked 7 years, 5 months ago Active 5 months ago Viewed 28k times

22

13

Given a kernel stack trace as below, how do you determine the specific line of code where the issue occurred?

```
kernel:  [<ffffffff80009a14>] __link_path_walk+0x173/0xfb9
kernel:  [<ffffffff8002cbec>] mntput_no_expire+0x19/0x89
kernel:  [<ffffffff8000eb94>] link_path_walk+0xa6/0xb2
kernel:  [<ffffffff80063c4f>] __mutex_lock_slowpath+0x60/0x9b
kernel:  [<ffffffff800238de>] __path_lookup_intent_open+0x56/0x97
kernel:  [<ffffffff80063c99>] .text.lock.mutex+0xf/0x14
kernel:  [<ffffffff8001b222>] open_namei+0xea/0x712
kernel:  [<ffffffff8006723e>] do_page_fault+0x4fe/0x874
kernel:  [<ffffffff80027660>] do_filp_open+0x1c/0x38
kernel:  [<ffffffff8001a061>] do_sys_open+0x44/0xbe
kernel:  [<ffffffff8005d28d>] tracesys+0xd5/0xe0
```

While I have no trouble finding the function call -- but translating __link_path_walk plus an offset to an actual line number is the difficult part.

Assuming this is for a standard distribution-provided kernel for which I know the exact version and build number, what's the process for fetching the necessary metadata and doing the corresponding lookup?

linux kernel debugging

Share Improve this question Follow

asked Jun 18 '14 at 6:08

tylerl
14.7k ● 7 ● 48 ● 71

5 Answers

Active Oldest Votes

14

Given an unstripped `vmlinux` with debugging symbols (typically included with "linux-devel" or "linux-headers" packages matching your kernel version), you can use the `addr2line` program included with binutils to translate addresses to lines in source files.

Consider this call trace:

```
Call Trace:
 [<ffffffff8107bf5d>] ? finish_task_switch+0x3d/0x120
 [<ffffffff815f3130>] __schedule+0x3b0/0x9d0
 [<ffffffff815f3779>] schedule+0x29/0x70
 [<ffffffff815f2ccc>] schedule_hrtimeout_range_clock.part.24+0xdc/0xf0
 [<ffffffff81076440>] ? hrtimer_get_res+0x50/0x50
 [<ffffffff815f2c6f>] ? schedule_hrtimeout_range_clock.part.24+0x7f/0xf0
 [<ffffffff815f2cf9>] schedule_hrtimeout_range_clock+0x19/0x60
 [<ffffffff815f2d53>] schedule_hrtimeout_range+0x13/0x20
 [<ffffffff811a8aa9>] poll_schedule_timeout+0x49/0x70
 [<ffffffff811aa203>] do_sys_poll+0x423/0x550
 [<ffffffff814eaf8c>] ? sock_recvmsg+0x9c/0xd0
 [<ffffffff811a8c50>] ? poll_select_copy_remaining+0x140/0x140
 [<ffffffff811a8c50>] ? poll_select_copy_remaining+0x140/0x140
 [<ffffffff811a8c50>] ? poll_select_copy_remaining+0x140/0x140
 [<ffffffff811a8c50>] ? poll_select_copy_remaining+0x140/0x140
 [<ffffffff811a8c50>] ? poll_select_copy_remaining+0x140/0x140
 [<ffffffff811a8c50>] ? poll_select_copy_remaining+0x140/0x140
 [<ffffffff811a8c50>] ? poll_select_copy_remaining+0x140/0x140
 [<ffffffff811a8c50>] ? poll_select_copy_remaining+0x140/0x140
 [<ffffffff811a8c50>] ? poll_select_copy_remaining+0x140/0x140
 [<ffffffff811aa3fe>] SyS_poll+0x5e/0x100
 [<ffffffff816015d2>] system_call_fastpath+0x16/0x1b
```

Then the address of the caller in `poll_select_copy_remaining` can be found with:

```
$ addr2line -e /tmp/vmlinux ffffffff811a8c50
/tmp/linux-3.15-rc8/fs/select.c:209
```

Share Improve this answer Follow

answered Jun 18 '14 at 14:02

Lekensteyn

5,901 ●5 ●36 ●54

2 My panic stack traces don't show the full addresses, only func+line. Is there a configuration to enable it? – Ciro Santilli 新疆再教育营六四事件法轮功郝海东 Apr 14 '18 at 22:12

1 @CiroSantilli包子露宪六四事件法轮功 Perhaps you could post a new question about that? You have to include more information: architecture (x86-64/ARM/...), Linux distribution, kernel version, and a sample of the output that does not match what you would expect. – Lekensteyn Apr 15 '18 at 8:46

3 OK, symbols only show when you have `CONFIG_KALLSYMS` : github.com/cirosantilli/linux-kernel-module-cheat/tree/... – Ciro Santilli 新疆再教育营六四事件法轮功郝海东 May 2 '18 at 17:05

1 @CiroSantilli新疆棉花TRUMPBANBAD the other answer here that is about using `faddr2line` works for `func+offset` , so no need to enable or recompile anything. – Hi-Angel May 16 at 11:43

17

I don't have a ~= RHEL5 at hand, so the output shown is from a Fedora 20, though the process should be mostly the same (the name of the function has changed).

You'd need to install the appropriate `kernel-debug-debuginfo` package for your kernel (assuming RHEL or derivative distro). This package provides a `vmlinux` image (an uncompressed not stripped version of the kernel):

```
# rpm -ql kernel-debug-debuginfo | grep vmlinux
/usr/lib/debug/lib/modules/3.14.7-200.fc20.x86_64+debug/vmlinux
```

that image can be used directly with `gdb`

```
# gdb /usr/lib/debug/lib/modules/3.14.7-200.fc20.x86_64+debug/vmlinux
GNU gdb (GDB) Fedora 7.7.1-13.fc20
Copyright (C) 2014 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
...
Reading symbols from /usr/lib/debug/lib/modules/3.14.7-
200.fc20.x86_64+debug/vmlinux...done.
(gdb) disassemble link_path_walk
Dump of assembler code for function link_path_walk:
   0xffffffff81243d50 <+0>:     callq  0xffffffff817ea840 <__fentry__>
   0xffffffff81243d55 <+5>:     push   %rbp
   0xffffffff81243d56 <+6>:     mov    %rsp,%rbp
   0xffffffff81243d59 <+9>:     push   %r15
   0xffffffff81243d5b <+11>:    mov    %rsi,%r15
   0xffffffff81243d5e <+14>:    push   %r14
   0xffffffff81243d60 <+16>:    push   %r13
   0xffffffff81243d62 <+18>:    push   %r12
   0xffffffff81243d64 <+20>:    push   %rbx
   0xffffffff81243d65 <+21>:    mov    %rdi,%rbx
   0xffffffff81243d68 <+24>:    sub    $0x78,%rsp
   0xffffffff81243d6c <+28>:    mov    %gs:0x28,%rax
   0xffffffff81243d75 <+37>:    mov    %rax,0x70(%rsp)
   0xffffffff81243d7a <+42>:    xor    %eax,%eax
   0xffffffff81243d7c <+44>:    movzbl (%rdi),%eax
   0xffffffff81243d7f <+47>:    cmp    $0x2f,%al
   ....
```

You can also use `objdump(1)` on the `vmlinux` image:

```
# objdump -rDlS /usr/lib/debug/lib/modules/3.14.7-200.fc20.x86_64+debug/vmlinux >
vmlinux.out
```

The flags are:

```
   -D
   --disassemble-all
       Like -d, but disassemble the contents of all sections, not just those expected
to contain instructions.
   -r
   --reloc
       Print the relocation entries of the file.  If used with -d or -D, the
relocations are printed interspersed with the
       disassembly.
   -S
   --source
       Display source code intermixed with disassembly, if possible.  Implies -d.
```

```
   -l
   --line-numbers
       Label the display (using debugging information) with the filename and source
line numbers corresponding to the object
       code or relocs shown.  Only useful with -d, -D, or -r.
```

You can lookup the function there:

```
ffffffff81243d50 <link_path_walk>:
link_path_walk():
/usr/src/debug/kernel-3.14.fc20/linux-3.14.7-200.fc20.x86_64/fs/namei.c:1729
 *
 * Returns 0 and nd will have valid dentry and mnt on success.
 * Returns error and drops reference to input namei data on failure.
 */
static int link_path_walk(const char *name, struct nameidata *nd)
{
ffffffff81243d50:       e8 eb 6a 5a 00          callq  ffffffff817ea840
<__entry_text_start>
ffffffff81243d55:       55                      push   %rbp
ffffffff81243d56:       48 89 e5                mov    %rsp,%rbp
ffffffff81243d59:       41 57                   push   %r15
ffffffff81243d5b:       49 89 f7                mov    %rsi,%r15
ffffffff81243d5e:       41 56                   push   %r14
ffffffff81243d60:       41 55                   push   %r13
ffffffff81243d62:       41 54                   push   %r12
ffffffff81243d64:       53                      push   %rbx
ffffffff81243d65:       48 89 fb                mov    %rdi,%rbx
ffffffff81243d68:       48 83 ec 78             sub    $0x78,%rsp
ffffffff81243d6c:       65 48 8b 04 25 28 00    mov    %gs:0x28,%rax
ffffffff81243d73:       00 00
ffffffff81243d75:       48 89 44 24 70          mov    %rax,0x70(%rsp)
ffffffff81243d7a:       31 c0                   xor    %eax,%eax
/usr/src/debug/kernel-3.14.fc20/linux-3.14.7-200.fc20.x86_64/fs/namei.c:1733
        struct path next;
        int err;

        while (*name=='/')
ffffffff81243d7c:       0f b6 07                movzbl (%rdi),%eax
ffffffff81243d7f:       3c 2f                   cmp    $0x2f,%al
ffffffff81243d81:       75 10                   jne    ffffffff81243d93
<link_path_walk+0x43>
ffffffff81243d83:       0f 1f 44 00 00          nopl   0x0(%rax,%rax,1)
/usr/src/debug/kernel-3.14.fc20/linux-3.14.7-200.fc20.x86_64/fs/namei.c:1734
                name++;
ffffffff81243d88:       48 83 c3 01             add    $0x1,%rbx
/usr/src/debug/kernel-3.14.fc20/linux-3.14.7-200.fc20.x86_64/fs/namei.c:1733
```

and match the offset to the actual line of code.

Share Improve this answer Follow

answered Jun 18 '14 at 10:17

dawud

14.7k ● 3 ● 40 ● 61

8

1. Install kernel-debuginfo
2. Download decode_stacktrace.sh which is in the kernel source tree.
3. Make stack dump output useful again.

```
 # ./decode_stacktrace.sh /usr/lib/debug/lib/modules/`uname -r`/vmlinux
/usr/lib/debug/lib/modules/4.1.12-112.14.14.el7uek.x86_64/ < ./trace > out
```

```
 # cat out
 [102820.087367] Call Trace:
 [102820.087371] dump_stack (/usr/src/debug/kernel-4.1.12/linux-4.1.12-
112.14.14.el7uek/lib/dump_stack.c:53)
 [102820.087375] warn_slowpath_common (/usr/src/debug/kernel-4.1.12/linux-4.1.12-
112.14.14.el7uek/kernel/panic.c:499)
 [102820.087378] warn_slowpath_null (/usr/src/debug/kernel-4.1.12/linux-4.1.12-
112.14.14.el7uek/kernel/panic.c:533)
 [102820.087380] af_alg_accept (/usr/src/debug/kernel-4.1.12/linux-4.1.12-
112.14.14.el7uek/include/net/sock.h:1689 /usr/src/debug/kernel-4.1.12/linux-
4.1.12-112.14.14.el7uek/crypto/af_alg.c:287)
 [102820.087382] alg_accept (/usr/src/debug/kernel-4.1.12/linux-4.1.12-
112.14.14.el7uek/crypto/af_alg.c:326)
 [102820.087385] SYSC_accept4 (/usr/src/debug/kernel-4.1.12/linux-4.1.12-
112.14.14.el7uek/net/socket.c:1485)
 [102820.087388] ? release_sock (/usr/src/debug/kernel-4.1.12/linux-4.1.12-
112.14.14.el7uek/net/core/sock.c:2415)
 [102820.087390] ? alg_setsockopt (/usr/src/debug/kernel-4.1.12/linux-4.1.12-
112.14.14.el7uek/crypto/af_alg.c:264)
 [102820.087393] SyS_accept (/usr/src/debug/kernel-4.1.12/linux-4.1.12-
112.14.14.el7uek/net/socket.c:1515)
 [102820.087395] system_call_fastpath (/usr/src/debug///////kernel-4.1.12/linux-
4.1.12-112.14.14.el7uek/arch/x86/kernel/entry_64.S:277)
 [102820.087397] ---[ end trace 1315ff0b8d6ff7d8 ]---
```

4. For a handful of function offsets, try `faddr2line` which is also in the kernel source.

```
 $ wget
https://raw.githubusercontent.com/torvalds/linux/master/scripts/faddr2line
 $ bash faddr2line /usr/lib/debug/lib/modules/`uname -r`/vmlinux
__do_softirq+0x92/0x320
 __do_softirq+0x92/0x320:
 ffs at arch/x86/include/asm/bitops.h:410
 (inlined by) __do_softirq at kernel/softirq.c:261
```

Share Improve this answer Follow edited Jun 28 '20 at 14:26

answered Mar 14 '18 at 6:24

wenjianhn

211 ●2 ●3

+1 for `faddr2line` as it works with kernel modules too, like for example: `faddr2line /lib/modules/5.12.0-arch1-1/build/drivers/gpu/drm/i915/i915.ko i915_gem_prime_import+0x2c/0x12`. The `decode_stacktrace.sh` I did not manage to make work: whatever path I give it, it always refuses to decode addresses in modules with the exact same complaint `readelf: Error: Not an ELF file - it has the wrong magic bytes at the start WARNING! Modules path isn't set, but is needed to parse this symbol` – Hi-Angel May 16 at 11:38

1

If addr2line should print a question mark for line number or objdump fails to inline source code and you have a custom kernel, be sure to recompile the kernel with CONFIG_DEBUG_INFO set. You might need to reproduce the error with the kernel just built.

Share Improve this answer Follow

answered Oct 5 '16 at 17:08

Johannes Thoma

111 ●3

With gdb, you can also use this command to find the line number quickly:

0

(gdb) list *(some_function+0x12c)

Share Improve this answer Follow

answered Jun 15 at 3:03

accessory

101 ● 2