Linux Perf Tools

Probe & Trace

Arnaldo Carvalho de Melo

Red Hat Inc.

January 30, 2015

Overview

- probe & trace
- Alternatives to existing tools
- But with new features!
- Using (relatively) recent kernel infrastructures
- perf, ftrace, kprobes, uprobes
- More coming: eBPF

Overview

- Strenghts
- Limitations
- Missing features
- Future work

Overview

- Tool output examples
- As non root user as much as possible

trace

- strace
- no ptrace: lower overhead
- doesn't stops the target
- Syscall argument prettifier
- More targets: syswide, cpu, others
- record/report workflow

Permissions

```
[acme@mica ~]$ ps ax|grep git
 3328 pts/0    S+     0:00 git remote update clark
 3329 pts/0    S+     0:00 git fetch --multiple clark
 3330 pts/0    S+     0:04 git fetch --append clark
 3370 pts/1    S+     0:00 grep git
[acme@mica ~]$ trace -p 3330
Error: Unable to find debugfs
Hint: Was your kernel compiled with debugfs support?
Hint: Is the debugfs filesystem mounted?
Hint: Try 'sudo mount -t debugfs nodev /sys/kernel/debug'
```

Permissions

```
[acme@mica ~]$ sudo mount -t debugfs nodev /sys/kernel/debug
[acme@mica ~]$ trace -p 3330
Error: No permissions to read /sys/kernel/debug/tracing/events/raw_syscalls/*
Hint: Try 'sudo mount -o remount,mode=755 /sys/kernel/debug'
```

Permissions

```
[acme@mica ~]$ sudo mount -o remount,mode=755 /sys/kernel/debug
[acme@mica ~]$ trace -p 3330
   0.012 (  0.006 ms): read(fd: 30<socket:[849]>, buf: 0x7215a0, count: 4      ) = 4
   0.070 (  0.017 ms): write(fd: 31<socket:[849]>, buf: 0x11e12c0, count: 1604 ) = 1604
 267.908 (267.836 ms): read(fd: 30<socket:[849]>, buf: 0x7fff503efb20, count: 4) = 4
 267.914 (  0.004 ms): read(fd: 30<socket:[849]>, buf: 0x7215a0, count: 4      ) = 4
 267.945 (  0.011 ms): write(fd: 31<socket:[849]>, buf: 0x11e12c0, count: 1604 ) = 1604
 533.696 (265.749 ms): read(fd: 30<socket:[849]>, buf: 0x7fff503efb20, count: 4) = 4
 533.701 (  0.003 ms): read(fd: 30<socket:[849]>, buf: 0x7215a0, count: 4      ) = 4
 533.800 (  0.013 ms): write(fd: 31<socket:[849]>, buf: 0x11e12c0, count: 1604 ) = 1604
^C
#
```

System wide tracing for non root user

```
[acme@mica ~]$ trace
Error: Operation not permitted.
Hint: Check /proc/sys/kernel/perf_event_paranoid setting.
Hint: For system wide tracing it needs to be set to -1.
Hint: Try: 'sudo sh -c "echo -1 > /proc/sys/kernel/perf_event_paranoid"'
Hint: The current value is 1.
[acme@mica ~]$
```

System wide tracing for non root user

```
[acme@mica ~]$ sudo sh -c "echo -1 > /proc/sys/kernel/perf_event_paranoid"
[acme@mica ~]$ trace  | egrep -v trace\|grep | head -10
 243.634 (0.000 ms): qpidd/382 ... [continued]: futex()) = -1 ETIMEDOUT Connection timed out
 243.654 (0.004 ms): qpidd/382 futex(uaddr: 0x7cb578, op: WAKE|PRIV, val: 1) = 0
2243.629 (1999.962 ms): qpidd/382 futex(uaddr: 0x7cb5a4, op: WAIT_BITSET|PRIV|CLKRT,
                                        val: 3775, utime: 0x7ffaa7a7bc90,
                                      val3: 4294967295) = -1 ETIMEDOUT Connection timed out
2243.636 (0.002 ms): qpidd/382 futex(uaddr: 0x7cb578,
                                     op: WAKE|PRIV, val: 1) = 0
2249.243 (0.000 ms): hald-addon-acp/875  ... [continued]: nanosleep()) = 0
2249.401 (0.154 ms): hald-addon-acp/875 socket(family: LOCAL,
                                               type: STREAM) = 4
2249.424 (0.020 ms): hald-addon-acp/875 connect(fd: 4, uservaddr: 0x7fff8e632870,
                                          addrlen: 110) = -1 ENOENT No such file or directory
2249.435 (0.002 ms): hald-addon-acp/875 close(fd: 4        ) = 0
2249.453 (0.002 ms): hald-addon-acp/875 rt_sigprocmask(how: BLOCK,
                                                       nset: 0x7fff8e6326f0,
                                                       oset: 0x7fff8e632670,
                                                       sigsetsize: 8) = 0
2249.456 (0.002 ms): hald-addon-acp/875 rt_sigaction(sig: CHLD, oact: 0x7fff8e632480,
                                                     sigsetsize: 8) = 0
[acme@mica ~]$
```

record/report workflow

- Similar to 'perf record' + 'perf report'
- Collect in one machine, analyse in another
- Needs better error handling, like 'live mode'
- Running as non root user has several issues
- Maybe I will be able to fix it in time for devconf ;-)

record

```
[acme@ssdandy linux]$ trace record usleep
Neither raw_syscalls nor syscalls events exist.
[acme@ssdandy linux]$ trace usleep
Error: No permissions to read /sys/kernel/debug/tracing/events/raw_s
Hint: Try 'sudo mount -o remount,mode=755 /sys/kernel/debug'

$ sudo mount -o remount,mode=755 /sys/kernel/debug
$ trace record usleep
Permission error mapping pages.
Consider increasing /proc/sys/kernel/perf_event_mlock_kb,
or try again with a smaller value of -m/--mmap_pages.
(current value: 1024)
$ trace -e nanosleep usleep
  1.960 ( 0.058 ms): nanosleep(rqtp: 0x7fffd9d1e730) = 0
$
```

Argument beautifiers

- Similar to strace
- But some need more data collected in the kernel
- Use 'perf probe' to collect those!

perf probe

- Create, then activate
- Becomes a tracepoint
- Collects variables
- kernel: kprobes
- Userspace: uprobes

Where to insert a probe

```
<getname_flags@/usr/src/debug/kernel-3.17.fc20/linux-3.17.8-200.fc20.x86_64/fs/namei.c:0>
      0  getname_flags(const char __user *filename, int flags, int *empty)
      1  {
          struct filename *result, *err;
          int len;
          long max;
          char *kname;
<SNIP>
     25   len = strncpy_from_user(kname, filename, max);
     26   if (unlikely(len < 0)) {
     27   err = ERR_PTR(len);
          goto error;
          }
<SNIP>
     65   result->uptr = filename;
     66   result->aname = NULL;
          audit_getname(result);
          return result;

         error:
     71   final_putname(result);
     72   return err;
     73  }
```

Inserting the probe

```
# perf probe 'vfs_getname=getname_flags:65 pathname=filename:string
Added new event:
  probe:vfs_getname    (on getname_flags:65 with pathname=filename:s

You can now use it in all perf tools, such as:

perf record -e probe:vfs_getname -aR sleep 1

# perf probe --list
  probe:vfs_getname    (on getname_flags:65@fs/namei.c with pathname
```

Trying it

```
# perf record -e probe:vfs_getname touch My-File-Name
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0.016 MB perf.data (~716 samples)
# perf report
# perf script
 touch 880 [2] 0.565794: probe:vfs_getname: (ffffffff8120b573) pathn
 touch 880 [2] 0.565802: probe:vfs_getname: (ffffffff8120b573) pathn
 touch 880 [2] 0.565817: probe:vfs_getname: (ffffffff8120b573) pathn
 touch 880 [2] 0.566003: probe:vfs_getname: (ffffffff8120b573) pathn
 touch 880 [2] 0.566048: probe:vfs_getname: (ffffffff8120b573) pathn
#
```

Back to trace

- trace uses this "vfs_getname" wannabe tracepoint if available
- "wannabe tracepoints" can be prototyped in this way
- Eventually some may become real tracepoints
- Changes in kernels may be isolated via a standard interface
- Location/variable name may change
- But "vfs_getname" and "pathname" remains

trace using vfs_getname

```
# trace -e open,close,dup2 touch My-File-Name
 0.671 (0.005 ms): open(filename: 0x7face05a3048, flags: CLOEXEC) = 3
 0.680 (0.001 ms): close(fd: 3</etc/ld.so.cache>                  ) = 0
 0.695 (0.006 ms): open(filename: 0x7face07a35e7, flags: CLOEXEC) = 3
 0.736 (0.001 ms): close(fd: 3</lib64/libc.so.6>                  ) = 0
 0.929 (0.007 ms): open(filename: 0x7face03473d0, flags: CLOEXEC) = 3
 0.945 (0.001 ms): close(fd: 3</usr/lib/locale/locale-archive>    ) = 0
 0.987 (0.008 ms): open(filename: 0x7ffff311b652, flags: CREAT|NOCTTY|NONBLOCK|WRONLY, mode: 4
 0.991 (0.002 ms): dup2(oldfd: 3<My-File-Name>                    ) = 0
 0.995 (0.001 ms): close(fd: 3<My-File-Name>                      ) = 0
 1.015 (0.001 ms): close(                                         ) = 0
 1.027 (0.002 ms): close(fd: 1                                    ) = 0
 1.031 (0.001 ms): close(fd: 2                                    ) = 0
#
```

trace needs more help from probe

- When copying syscall arguments from userspace to the kernel
- How to signal how much to copy?
- Automatically creating the probes?

probe integration with other tools

- Probes are just tracepoints
- All other features presents: callchains
- Quick scripting in perl, python

The life of a ping packet

```
$ perf probe -L icmp_rcv
<icmp_rcv@/usr/src/debug/kernel-3.17.fc20/linux-3.17.8-200.fc20.x86_64/net/ipv4/icmp.c:0>
      0  int icmp_rcv(struct sk_buff *skb)
      1  {
           struct icmphdr *icmph;
      3    struct rtable *rt = skb_rtable(skb);
           struct net *net = dev_net(rt->dst.dev);
<SNIP>
     28    if (skb_checksum_simple_validate(skb))
           goto csum_error;

     31    if (!pskb_pull(skb, sizeof(*icmph)))
           goto error;

     34    icmph = icmp_hdr(skb);
<SNIP>
     51    if (rt->rt_flags & (RTCF_BROADCAST | RTCF_MULTICAST)) {
           /*
            * RFC 1122: 3.2.2.6 An ICMP_ECHO to broadcast MAY be
            *   silently ignored (we let user decide with a sysctl).
            * RFC 1122: 3.2.2.8 An ICMP_TIMESTAMP MAY be silently
            *   discarded if to broadcast/multicast.
            */
     58    if ((icmph->type == ICMP_ECHO ||
     59         icmph->type == ICMP_TIMESTAMP) &&
```

Source code view

- Shows line number offsets
- Where probes can be inserted
- Requires DWARF information
- Matching debuginfo packages or 'gcc -g' built binaries

TO BE CONTINUED

- will use perf probe -g python
- show collecting callchains
- tried all the way to ping.c main()
- but it failed at the 1st userspace IP
- will try with a newer kernel
- this was with 3.17.8-200.fc20.x86_64

That is all folks!

Thanks!

Arnaldo Carvalho de Melo

acme@kernel.org

acme@redhat.com

linux-perf-users@vger.kernel.org