FreeBSD ABI: Shared Page

Константин Белоусов
kib@freebsd.org

29 сентября 2012 г.

Revision : 1.8

ABI

ABIs supported by FreeBSD/amd64

1. FreeBSD 64bit ELF (AKA amd64)
2. FreeBSD 32bit ELF (AKA i386)
3. FreeBSD 32bit a.out (AKA FreeBSD 1.x/2.x)
4. Linux 32bit ELF

What is the shared page

1. Single page, forcibly inserted into the process VA on image activation.
2. Shared by all images with the same ABI, no process-private data.
3. R/O for usermode.
4. Used by several subsystems. Kernel provides suballocator.

Process memory layout

```
pooma% procstat -v 112
  PID START      END        PRT  FL    TP PATH
112 0x8048000  0x8049000  r-x CN-- vn /bin/sleep
112 0x8049000  0x804a000  rw- ---- df
...
112 0x28097000 0x28196000 r-x CN-- vn libc.so.7
112 0x28196000 0x2819c000 rw- C--- vn libc.so.7
112 0x2819c000 0x281b3000 rw- ---- df
112 0xfffde000 0xffffe000 rwx ---D df
112 0xffffe000 0xfffff000 r-x CN-- ph
```

Shared page uses: Non-Executable Stacks

"Security"

- An attempt to make the shell code drop less trivial.
- Supported by other OSes.
- Some ABIs specify that stacks are -x from inception.

NX Stacks: GNU ELF Extension

- Extension to the ELF
- Works on FreeBSD/amd64 (32 and 64 bit), FreeBSD/powerpc (32 and 64 bit), FreeBSD/i386 PAE.

Signal trampolines

```
NON_GPROF_ENTRY(sigcode)
    call *SIGF_HANDLER(%rsp) /* call signal handler
    lea SIGF_UC(%rsp),%rdi /* get ucontext_t */
    pushq $0            /* junk to fake return addr. */
    movq  $SYS_sigreturn,%rax
    syscall             /* enter kernel with args */
0:  hlt             /* trap priviliged instruction */
    jmp 0b
```

How to verify required stack protection mode

PT GNU STACK

```
pooma% readelf -l /usr/lib32/libc.so.7
Elf file type is DYN (Shared object file)
Entry point 0x20f90
There are 6 program headers,
     starting at offset 52
Program Headers:
  Type           VirtAddr   MemSiz    Flg  Align
  LOAD           0x00000000 0x10c31c  R E  0x1000
  LOAD           0x0010d31c 0x1c8ac   RW   0x1000
  DYNAMIC        0x0010f4a8 0x000c8   RW   0x4
  TLS            0x0010d31c 0x00014   R    0x4
  GNU_EH_FRAME   0x0010bee4 0x000e4   R    0x4
  GNU_STACK      0x00000000 0x00000   RW   0x4
```

How to specify required stack protection mode

C Compiler

- Automatic, only uses +x when generating trampolines.

Assembler

- in source:

```
.section .note.GNU-stack,"",%progbits
```

- on the command line:

```
as --[no]execstack
```

NX Stack: Implementation

Kernel

- Elf Image Activator parses `PT_GNU_STACK` and creates initial stack with the right protection
- auxv `AT_STACKPROT` for rtld

Rtld

- shared objects `PT_GNU_STACK` segments

libc and libthr

- `__pthread_map_stacks_exec` callback, called from rtld

Shared page uses 2: Fast gettimeofday(2)

The problem

- FreeBSD `gettimeofday(2)` is very precise but slow.
- Naive programs calls `gettimeofday(2)` too often.

Why slow ?

1. Syscall.
2. Precise.

Why slow ?

Timecounters

- RDTSC
- HPET

Timehands

```
struct vdso_timehands {
    uint32_t th_algo;
    uint32_t th_gen;
    uint64_t th_scale;
    uint32_t th_offset_count;
    uint32_t th_counter_mask;
    struct bintime th_offset;
    struct bintime th_boottime;
    VDSO_TIMEHANDS_MD
};
```

Gettimeofday implementation

timecounter read

```
static u_int
tc_delta(const struct vdso_timehands *th)
{
 return ((__vdso_gettc(th) - th->th_offset_count) &
     th->th_counter_mask);
}
```

Gettimeofday implementation (cont)

binuptime

```
static int binuptime(struct bintime *bt,
  struct vdso_timekeep *tk, int abs)
{
  struct vdso_timehands *th;
...
 {
    curr = tk->tk_current;
    rmb();
    th = &tk->tk_th[curr];
    *bt = th->th_offset;
    bintime_addx(bt, th->th_scale * tc_delta(th));
    if (abs)
       bintime_add(bt, &th->th_boottime);
 }
}
```

Gettimeofday implementation (cont 2)

Usermode implementation of gettimeofday(2)

- timehands = * auxv `AT_TIMEKEEP`
- `__vdso_gettc` == `RDTSC`

Measured speedup of gettimeofday(2)

- Nehalem (i7 930): $\sim$ 4x
- SandyBridge (i7 2600K): $\sim$ 7x

Errands

TODO: VDSO

- Signal trampolines unwind.
- Syscall overrides using rtld symbolic interposition.
- No libc changes required.

Trivia

- FreeBSD-SA-12:04.sysret