Program supervision with Ptrace
Agenda

- What is Ptrace
- How/Where Ptrace is implemented
- How to relate this to HW4
- How to use ptrace for program supervision
  - Writing a strace-clone (ish): snyfer
- Warning: all code refers to Linux kernel 2.6.27.41

Acknowledgement: Code “snyfer” provided by Prof. Michael Locasto
What is Ptrace?

- **Process Trace** (man 2 ptrace)
- System call that allows a process to monitor and control the execution of a child process
- Two basic methods:
  - Parent performs a *fork* and the child process announces it wants to be traced by requesting: “TRACEME!”
  - Process requests to ATTACH to another process (becoming its parent)
- But.... how can one perform these “requests”?
Ptrace Requests

```c
#include <sys/ptrace.h>

long ptrace(enum __ptrace_request request, pid_t pid, void *addr, void *data);
```

- **PTRACE_TRACEME**: Called by the child that wants to be traced. All signals (except SIGKILL) make the child to stop and its parent decides how to proceed.
- **PTRACE_ATTACH**: A process wants to control another one, becoming its parent.
- **PTRACE_PEEKUSER**: Read user data information (e.g. registers).
- **PTRACE_SYSCALL**: Resume execution and stops at the next system call.
- **PTRACE_DETACH**: Stops tracing. Restores information.
- **Others**: See ptrace manual page (man 2 ptrace).
Ptrace implementation

#include <sys/ptrace.h>

long ptrace(enum __ptrace_request request, pid_t pid, 
            void *addr, void *data);

- Peeking into the kernel
  - task_struct: e.g., ptrace, parent & real_parent
  - Defining system call
    - syscalls.h, syscall_table_32.S
  - Actual implementation in two logical parts:
    - Abstract/General
    - Architecture-specific
General implementation

- **Ptrace API**
  - Defines function prototypes and “types” for ptrace requests
  - `$WORK/include/linux/ptrace.h`

- **Ptrace Implementation**
  - Implements system call (SYSCALL_DEFINE4)
  - “Higher” level implementation (e.g., find process, attach, “traceme”)
  - `$WORK/kernel/ptrace.c`
Architecture Specific Ptrace

- **Platform-specific API**
  - e.g., $WORK/include/asm-x86/ptrace.h

- **Architecture-specific ABI**
  - e.g., $WORK/include/asm-x86/ptrace-abi.h
  - Application Binary Interface: describes low level interface between program and OS

- **Architecture-specific Implementation**
  - e.g., $WORK/arch/x86/kernel/ptrace.c
What about homework 4?

Task 1. DONTTRACEME (40 points)

The Mac OS X implementation of the ptrace(2) mechanism provides the PT_DENY_ATTACH flag. The manual page states that the semantics of PT_DENY_ATTACH are:

"This request is the other operation used by the traced process; it allows a process that is not currently being traced to deny future traces by its parent. All other arguments are ignored. If the process is currently being traced, it will exit with the exit status of ENOTSUP; otherwise, it sets a flag that denies future traces. An attempt by the parent to trace a process which has set this flag will result in a segmentation violation in the parent."

Implement this feature in the Linux kernel 2.6.27.41...

- What do you need to do?
  - Request PTRACE_DONTTRACEME to set flag PT_DENY_ATTACH
  - Check for error (ENOTSUP)
  - Change attaching procedure to check this flag, send segfault signal
Let's practice!

Snyfing processes...