x64 assembly programming using system calls

Arvind S Raj
(arvindsraj@am.amrita.edu)

CSE467 IntroSSOC

B.Tech CSE Jan-May 2017

Looking back and ahead

Till now

- x64 ASM programming using C library.
- Debugger-fu and ltrace-fu for improved ASM programming experience.

Up next

- x64 ASM programming without the C library: straight to Linux kernel API!
- More debugging tools to help debug kernel API calls.

Recap from Operating Systems

- What is the kernel?
- What are system calls?

Recap from Operating Systems(cont.)

- **Kernel**: Core system software that provides all software controlled access to available hardware.
- **System calls**: Functions that can be used to avail services provided by the kernel.
- Almost any action that affects a hardware peripheral(network card, monitor, keyboard, headset) is done by kernel, when invoked using a system call.

System calls in Linux kernel

- Vary between 32 bit and 64 bit kernels.
- System call ID, argument passing and invocation differ.
- Counts also vary: more in 32 bit last I checked.
- Well documented in multiple places so no need to memorize.
- About 300 system calls provided by 64 bit Linux kernel.

System call vs C library

- System call lower level than C library.
- Most C library functions: thin wrapper around or composition of system calls.
- Lower level $\implies$ more details to handle.
- Usually never invoked directly by application software.

Why learn programming with system calls?

- Why would anyone invoke system calls directly? Seems frustrating!
- Exploitation: C library won't be always available or desired functions won't be.
- Easier to invoke system calls than discover library functions.
- Also useful in RE to understand unknown binary's behaviour: access to hardware possible via kernel only.

Steps for invoking a 64 bit system call

- Store system call number in RAX.
- Store arguments in other general purpose registers: semantics vary between system calls.
- Invoke system call using *syscall* instruction.
- Return value of system call in RAX register.

Sample programs!

Let's go through sample programs to learn assembly programming using system calls.

Debugging system call invocations

- GDB and pwndbg are still useful for inspecting program state.
- **strace**: ltrace equivalent for inspecting system calls.
- Prints out the system call invoked and the corresponding arguments.
- Also displays any OS signals received.
- Accepts wide variety of options: read man page to know what's supported.

System calls: summary

- System calls invoke kernel directly.
- More likely to be available than C library functions.
- System call number in RAX, arguments in other GPRs. Read documentation for more details.
- Use strace, GDB and pwndbg for debugging when things go wrong.