COP4600 Operating Systems

Exercise 3 Code Reconnaissance

Johnny Li

2/4/2020

The printmypid.c program fetch and print the pid of the current process. This process begins in the program with the function “getpid()” which call the user-space library that goes into the kernel to handle the system call function to get the process id. Prior to entering the kernel mode, the program includes the unistd.h which declared functions and defined macros that comprise the POSIX operating system API.

When the function “getpid()” is call there is then a searched on the unistd.h which is the user-space library, located at src/linux-hwe-5.0.0/usr/include/asm-generic/unistd.h, to obtain the declaration of the function.

Shown at line 522:

#define \_\_NR\_getpid 172

\_\_SYSCALL(\_\_NR\_getpid, sys\_getpid)

This triggers the execution of the system call with the parameter “\_\_NR\_getpid” and “sys\_getpid”, which ultimately start the system call for getpid. The “\_\_NR\_getpid” is the system call number associated with getpid since processes does not refer to system calls by name, but by their system call number. A system call number is an argument that a process must pass to the kernel in order to identify the requested system call. The kernel associate system call number with its service, the process then accesses the call dispatch table to obtain the number, which is stored in the sys\_call\_table array.

This is at /src/linux-hwe-5.0.0/arch/x86/entry/syscalls/syscall\_64.tbl. The system call number is retrieved, being at number 39 for getpid at line 50 which is then pass as a parameter to identify the system call into the function.

The system call function gets define at /src/linux-hwe-5.0.0/kernel/sys.c line 891.

SYSCALL\_DEFINE0(getpid){

return task\_tgid\_vnr(current);

}

Note that this returns the tgid not the pid which is identical unless CLONE\_THREAD was specified. The task\_tgid\_vnr(current) function, where the parameter is a pointer to store the location of the pid/tgid, is defined in /src/linux-hwe-5.0.0/include/linux/sched.h at line 1329:

static inline pid\_t task\_tgid\_vnr(struct task\_struct \*tsk){

return \_\_task\_pid\_nr\_ns(tsk, PIDTYPE\_PID, NULL);

}

This function takes in the task structure pointer, or process pointer, in order to refer the variable values of the process as a parameter for another function called \_\_task\_pid\_nr\_ns(). The \_\_task\_pid\_nr\_ns() function, task pid number and name space, is where the pid is actually obtained. This \_\_task\_pid\_nr\_ns() is located at /src/linux-hwe-5.0.0/include/linux/sched.h line 1386:

static inline [pid\_t](https://elixir.bootlin.com/linux/latest/ident/pid_t) [task\_ppid\_nr\_ns](https://elixir.bootlin.com/linux/latest/ident/task_ppid_nr_ns)(const struct [task\_struct](https://elixir.bootlin.com/linux/latest/ident/task_struct) \*[tsk](https://elixir.bootlin.com/linux/latest/ident/tsk), struct [pid\_namespace](https://elixir.bootlin.com/linux/latest/ident/pid_namespace) \*ns){

[pid\_t](https://elixir.bootlin.com/linux/latest/ident/pid_t) [pid](https://elixir.bootlin.com/linux/latest/ident/pid) = 0;

[rcu\_read\_lock](https://elixir.bootlin.com/linux/latest/ident/rcu_read_lock)();

if ([pid\_alive](https://elixir.bootlin.com/linux/latest/ident/pid_alive)([tsk](https://elixir.bootlin.com/linux/latest/ident/tsk)))

[pid](https://elixir.bootlin.com/linux/latest/ident/pid) = [task\_tgid\_nr\_ns](https://elixir.bootlin.com/linux/latest/ident/task_tgid_nr_ns)([rcu\_dereference](https://elixir.bootlin.com/linux/latest/ident/rcu_dereference)([tsk](https://elixir.bootlin.com/linux/latest/ident/tsk)->real\_parent), ns);

[rcu\_read\_unlock](https://elixir.bootlin.com/linux/latest/ident/rcu_read_unlock)();

[return](https://elixir.bootlin.com/linux/latest/ident/return) [pid](https://elixir.bootlin.com/linux/latest/ident/pid);

}

As seen above, the [pid\_t](https://elixir.bootlin.com/linux/latest/ident/pid_t) [task\_ppid\_nr\_ns](https://elixir.bootlin.com/linux/latest/ident/task_ppid_nr_ns) function initialize return variable to be empty and then sets a read lock. The function then enters its critical section of reading the pid, if it exist, and afterwards unlock the read lock. The pid information is returned.

Upon obtaining the pid information from \_\_task\_pid\_nr\_ns, pid\_t task\_tgid\_vnr also return the pid information tied to the process structure, which is then handed off to the system call function of SYSCALL\_DEFINE0(getpid). At this point the system call is finished and leaves the kernel mode to enter user mode where the pid information is to be stored in the variable my\_pid for the user to work with.

Listed files and paths:

unistd.h: src/linux-hwe-5.0.0/usr/include/asm-generic/unistd.h line 522

syscall\_64.tbl: /src/linux-hwe-5.0.0/arch/x86/entry/syscalls/syscall\_64.tbl line 50

sys.c: /src/linux-hwe-5.0.0/kernel/sys.c line 891

sched.h: /src/linux-hwe-5.0.0/include/linux/sched.h line 1329 and 1386