P1 Report - Michael Schultes - 9/28/2018

Process Table

Every process under Linux is dynamically allocated a struct task_struct structure [1]. This is defined in the include/linux/sched.h file. So, to add a new attribute to all processes, an ‘int security_level’ was inserted just below the header inside task_struct.

Fork is the primary (and historically, only) method of process creation on Unix-like operating systems. This is defined in the kernel/fork.c file. So, to initialize the security level for all processes, the inside of the _do_fork() method was edited to include the line ‘p->security_level=0’, where ‘p’ is the pointer used to reference the task_struct for a process inside the function.

System Call

The SYSCALL_DEFINE[0-6] macro is used (obviously) to define a given block of code as a system call [2]. Many system calls are defined this way in the kernel/sys.c file. So, to define the new system calls, ‘get_security_level’ and ‘set_security_level’ were added to this file where they are defined with SYSCALL_DEFINE[] and given logic based on the requirements given in the P1 documentation.

The Linux kernel includes a file which lists each system call in a table. This file is processed by various scripts at build time to generate header files which can be used by user programs [2]. The system call table can be found in the arch/x86/entry/syscalls/syscall_64.tbl file. So, to define the system call numbers for the newly defined system calls, entries for both are added at the end of the list of 64bit system calls, call numbers 332 and 333, following the conventions of the other call numbers above them.

Static Library

Within the home directory a new securitylevel directory was created which contained the source and header files which store library function prototypes and harness test functions. A Makefile also exists to compile the source and create the libsecuritylevel.a file.

Testing

The getlevel.c and setlevel.c files were used to perform tests on the library functions within securitylevel. A bug does arise when using the getlevel program on a non-existent pid, which results in returning the system call number for the get_security_level system call, 332. The provided test files securitytest.c and harnesstest.c were compiled and run from the home directory (user space) with all tests passing.

References

1. Linux Kernel 2.4 Internals, Tigran Aivazian tigran@veritas.com, http://www.tldp.org/LDP/lki/, last accessed 9/28/2018
2. The Definitive Guide to Linux System Calls, Joe Damato, https://blog.packagecloud.io/eng/2016/04/05/the-definitive-guide-to-linux-system-calls/#what-is-a-system-call, last accessed 9/28/2018