Project 1: System Calls

SYNOPSIS
The goal of this project is to implement a system call in Reptilian in addition to three static library functions that

DESCRIPTION
syscall_64.tbl (/usr/rep/src/reptilian-kernel/arch/x86/entry/syscalls/syscall_64.tbl): Modified to register new syst
applications.

syscalls.h (/usr/rep/src/reptilian-kernel/include/linux/syscalls.h): Updated to declare the prototypes of the new sy

sys.c (/usr/rep/src/reptilian-kernel/kernel/sys.c): Implements the system calls, setting, getting, and logging the pr
within the kernel.

The process log level and message logging are managed through specific functions in my library.
The get_proc_log_level() retrieves the current log level using the GET_LOG_LEVEL system call.
The set_proc_log_level(int new_level) attempts to set a new log level through the SET_LOG_LEVEL system ca
which includes a kernel-side check to ensure the operation is performed only by a superuser (using this comman
returning the new level on success or -1 on failure. Both functions use syscall() to interact directly with the kerne

The kernel function SYSCALL_DEFINE2(proc_log_message, char*, message, int, level) is essential for logging
This function employs printk for logging since printk has efficient handling of log messages at various severity l
directing them to the kernel's log buffer, accessible via dmesg.
This method ensures that messages are only logged if they meet the current system's log level requirements.
The library function proc_log_message(int level, char *message) allows users make a system call (PROC_LOG_
level message in the the kernel's dmesg buffer if the set log level is appropriate.
It returns the log level on successful logging or -1 for invalid inputs.

To acquire the executable name and pid I used the "current" macro.
This macro provides access to the task_struct of the currently executing process.
In the task_struct, the comm field represents the executable name associated with the running process and the pi
The use of current->comm and current->pid directly in the syscall ensures that each log entry is clearly associate
invoking the system call.

TESTING
I applied my patch file (p1.diff) to a clean kernel and used the command: make && sudo make install && sudo
Then I rebooted the VM to ensure the changes I made such as adding system calls, etc. were applied.
To ensure the process_log.tar.gz packaging was done correctly, I created a static library using the Makefile. With
testing files and ran the executables to ensure my output matched the expected output.

BUGS
There are no known bugs.

LINK
Link to screencast

REFERENCES/CITATIONS
"Project 1: System Calls." University of Florida, 2023.

AUTHOR
Aarithi Rajendren

ReplyReply AllForward DeleteSpam