EXPERIMENTAÇÃO COM O KERNEL 3.16.0 DO LINUX

Introdução

Este experimento descreve os passos realizados na elaboração de um programa para instrumentalizar funções do Kernel 3.16.0 do sistema operacional Linux. Mais especificamente, o programa contabiliza quantas vezes uma determinada função do código do Kernel é utilizada por um processo (aplicação do sistema ou usuário). Para isto, o programa receberá como parâmetros: a identificação da aplicação (*Process Identification* – PID), a identificação de uma determinada função (representada por um índice de 0 à 22) e um valor 0 ou 1, para zerar ou ler o valor do contador, respectivamente.

O primeiro passo é identificar quais funções do Kernel seriam instrumentadas. A Tabela 1 apresenta as 23 funções escolhidas.

Função Instrumentada	Localização	Índice
tcp_sendmsg()	net/ipv4/tcp.c	0
kbd_event()	drivers/tty/vt/keyboard.c	1
do_fork()	kernel/fork.c	2
set_page_dirty()	mm/page-writeback.c	3
do_path_lookup()	fs/namei.c	4
__iget()	fs/inode.c	5
__ide_do_rw_disk()	drivers/ide/ide-disk.c	6
__alloc_pages_high_priority()	mm/page_alloc.c	7
__free_pages()	mm/page_alloc.c	8
request_dma()	kernel/dma.c	9
free_dma()	kernel/dma.c	10
sock_sendmsg()	net/socket.c	11
kernel_sendmsg()	net/socket.c	12
tcp_read_sock()	net/ipv4/tcp.c	13
tcp_recvmsg()	net/ipv4/tcp.c	14
tcp_init_sock()	net/ipv4/tcp.c	15
balance_dirty_pages()	mm/page-writeback.c	16
__bdi_update_bandwidth()	mm/page-writeback.c	17
add_wait_queue()	kernel/sched/wait.c	18
msleep()	kernel/timer.c	19
add_timer_on()	kernel/timer.c	20
put_pid()	kernel/pid.c	21
eth_header()	net/ethernet/eth.c	22

Alterações no Kernel

Primeiramente, é necessário inserir na estrutura de cada processo do sistema operacional um *buffer* de memória com 23 posições, correspondentes a cada uma das funções escolhidas. Para isto deve ser inserido um array de 23 posições dentro da estrutura *struct task_struct*.

Em seguida, devem ser inseridas as linhas de código abaixo em cada uma das funções instrumentadas, para o incremento do contador. Neste exemplo, o índice 0 do array significa que este código foi inserido na função *tcp_sendmsg()*.

```
struct task_struct *tarefa = current;
tarefa->contador[0]++;   /* incrementa o contador da funcao tcp_sendmsg() (indice 0) */
```

Para que o programa possa interagir com as funções do Kernel é necessária a criação de uma *system call*, que deve ser chamada de *scontador*. Esta *system call* deve receber o *pid* do processo que deseja-se consultar, o índice da função correspondente e a condição (0 ou 1), indicando se deseja zerar (0) ou ler (1) o conteúdo do buffer, e retornar o valor do buffer.

Teste da System Call

Após a compilação e inicialização do sistema com o Kernel modificado é necessário testar a *system call* através de um programa de usuário.

A utilização do programa é bem simples. Basta passar como parâmetros o PID de um processo do sistema, o índice da função que deseja-se verificar (de acordo com a Tabela 1) e um parâmetro para indicar se o programa deve ler o contador ou zerá-lo. Fica definido que este parâmetro deve ser 0 para zerar o contador e diferente de 0 para ler o contador. Dessa forma, a utilização do programa possui a seguinte sintaxe abaixo.

```
#./nome_do_programa [PID] [INDICE] [CONDICAO]
```

Exemplos de Resultados

No primeiro exemplo ilustrado abaixo, temos a verificação de quantas vezes o processo *init* (PID 1) utilizou a função do Kernel *do_fork()* (índice 2). Em seguida, foi zerado o contador desta função dentro do processo *init*.

```
debian@debian:/home/debian/experimento-so# ./a.out 1 2 1
Qtd = 72
debian@debian:/home/debian/experimento-so# ./a.out 1 2 0
Contador da funcao 2 zerado (0)
debian@debian:/home/debian/experimento-so# ./a.out 1 2 1
Qtd = 0
```

O próximo exemplo ilustra quantas vezes o interpretador de comandos *bash* utilizou a função *do_fork()*. Nota-se que, após zerado o contador desta função, o resultado é 1 porque já está contabilizando a utilização do programa *a.out*.

```
debian@debian:/home/debian/experimento-so# ps
PID    TTY    TIME        CMD
642    tty1   00:00:00    login
717    tty1   00:00:00    su
718    tty1   00:00:00    bash
```

893 tty1 00:00:00 ps
debian@debian:/home/debian/experimento-so# ./a.out 718 2 1
Qtd = 96
debian@debian:/home/debian/experimento-so# ./a.out 718 2 0
Contador da funcao 2 zerado (0)
debian@debian:/home/debian/experimento-so# ./a.out 718 2 1
Qtd = 1