

Relatório Projeto 3

Gabriel de Paula e Lima 587710
Giovana Vieira de Moraes 587591

2 de novembro de 2017

A atividade

- Compilar o módulo fornecido como exemplo
- Modificar o módulo fornecido para exibir, no lugar da frase fixa, o PID do processo lendo o arquivo e o PID do seu processo pai
- Dar ao interpretador de comando executando o processo de leitura permissões de root

Exibir o PID do processo lendo o arquivo (cat) e do processo pai (bash)

Para essa função, foi necessário usar a `task_struct`, que é uma struct que descreve informações de processos ou tarefas do sistema, guardando informações importantes como PID, nome do processo atual, credenciais do grupo e do processo e processo pai.

Após a declaração de um ponteiro para a estrutura, só foi necessário imprimir o nome (`task->comm`) e o pid (`task->pid`) do processo atual. Para imprimir as informações do processo pai: `task->parent->comm` e `task->parent->pid`.

Dar ao processo pai permissões de root

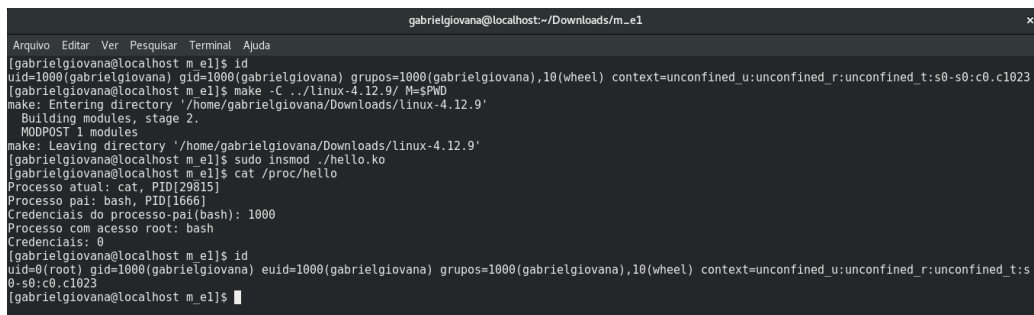
Assim como mencionado acima, a `task_struct` contém em suas informações as credenciais do usuário e do grupo ao qual o processo corrente pertence. Mais especificamente, essas credenciais são definidas dentro da estrutura de dados `cred`.

Para darmos permissão de root ao processo pai, nesse caso o (`bash`), precisamos alterar o `euid` que é a variável definidora do tipo de permissão do processo. Como não podemos alterá-la diretamente, então é necessário criar uma struct do tipo `cred`, que nos permitirá alterar as credenciais do processo.

Para pegar a permissão atual do processo, usamos a função `get_cred`, que retorna ao `cred` do processo um ponteiro editável por meio da struct criada por nós. Foi alterado o valor o `euid` para 0, que é o valor de permissão root e então foi usado `put_cred`, o qual sacramenta a mudança de permissão do processo para acesso root.

Dificuldades encontradas

Inicialmente tivemos dificuldade em entender o que nos era pedido para realizar, tais dúvidas foram sanadas por meio de questionamentos ao professor e leitura de manuais sobre as estruturas a serem alteradas para realizar o trabalho. Posteriormente tivemos dificuldade em definir se o `euid` ou o `uid` deveriam ser alterados a fim de dar permissão de root ao usuário, chegando a conclusão por meio de pesquisas de que o `euid` deveria ser alterado. Pois ele diz respeito as permissões do processo em si enquanto o `uid` diz respeito as permissões do usuário. Inclusive a primeira versão funcional do programa foi feita alterando o `uid` como mostra a imagem abaixo.



```
gabrielgiovana@localhost:~/Downloads/m_e1
Arquivo Editar Ver Pesquisar Terminal Ajuda
[gabrielgiovana@localhost m_e1]$ id
uid=1000(gabrielgiovana) gid=1000(gabrielgiovana) grupos=1000(gabrielgiovana),10(wheel) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[gabrielgiovana@localhost m_e1]$ make -C ../linux-4.12.9/ M=$PWD
make: Entering directory '/home/gabrielgiovana/Downloads/linux-4.12.9'
  Building modules, stage 2.
  MODPOST 1 modules
make: Leaving directory '/home/gabrielgiovana/Downloads/linux-4.12.9'
[gabrielgiovana@localhost m_e1]$ sudo insmod ./hello.ko
[gabrielgiovana@localhost m_e1]$ cat /proc/hello
Processo atual: cat, PID[29815]
Processo pai: bash, PID[1666]
Credenciais do processo-pai(bash): 1000
Processo com acesso root: bash
Credenciais: 0
[gabrielgiovana@localhost m_e1]$ id
uid=0(root) gid=1000(gabrielgiovana) euid=1000(gabrielgiovana) grupos=1000(gabrielgiovana),10(wheel) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[gabrielgiovana@localhost m_e1]$
```