Relatório do EP1 de MAC0422

Caio Rodrigues Gama (12543381)
Matheus Sanches Jurgensen (12542199)

2 de abril de 2023

1 Explicação do Código

A fim de solucionar o que foi proposto para este EP, o código foi implementado e modularizado por meio de algumas funções, as quais serão melhor explicadas a seguir.

1.1 Shell Loop

Para que a nossa shell funcione, foi implementada a função void shell_loop() que se utiliza de um loop *while* que roda indefinidamente até que o usuário escreva *quit* na shell. Essa função recebe qual o comando a ser executado, bem como o caminho do arquivo ou programa que o comando utilizará. Assim, a função do loop direciona a execução a um dos 4 comandos possíveis, descritos a seguir.

1.2 Primeiro Comando

Para o comando nem_eu_nem_de_ninguem <caminho do arquivo>, foi criada uma função void protection_000(char* address) que recebe o caminho do arquivo e modifica a proteção dele para 000, utilizando a chamada de sistema chmod, de tal forma que o arquivo passa a ter nenhuma permissão de acesso. Ao final, a função imprime uma mensagem que indica se a atualização de permissão foi ou não feita com êxito.

1.3 Segundo Comando

O comando soh_eumesmo <caminho do arquivo> funciona de forma similar ao anterior, também gerando alterações nas permissões do arquivo. Desta vez, no entanto, a função criada foi a void protection_700(char* address), que também recebe o caminho do arquivo e muda sua permissão de acesso para que somente o proprietário possa ler, escrever e executar. Também possui a mensagem final que indica se a mudança foi feita com sucesso ou não.

1.4 Terceiro Comando

A implementação do comando rodaeolhe <caminho do programa> foi feita utilizando a função int run_and_see(char* address) que usa a chamada fork() para criar um novo processo filho da seguinte forma: **pid_t pid = fork()**. A variável **pid** valerá 0 para o processo filho e um inteiro positivo para o processo pai. Se, por outro lado, for negativa, significa que a chamada fork() falhou. Caso **pid** for igual a 0, o código, no processo filho, executa o programa dado como parâmetro com execve(). No processo pai, a chamada waitpid() é utilizada para que ele aguarde o fim de seu processo filho para continuar sua execução. Quando terminar, o programa exibe uma mensagem explicitando com qual código de saída o processo filho finalizou a execução.

1.5 Quarto Comando

O comando sohroda <caminho do programa> utiliza a função int run_in_background(char* address) que inicia de forma semelhante à anterior, por meio da chamada fork(). Dessa vez, no entanto, a shell deve monopolizar o teclado, enquanto o processo filho faz a execução do programa em *background*.

A tela deve manter-se mostrando a saída da shell e do programa. Para fechar o acesso do processo filho ao teclado, usa-se um *pipe*, redirecionando-se a entrada padrão do filho (antes o teclado) para o *pipe*. Dessa forma, o teclado está monopolizado pela shell. De forma similar ao comando rodaeolhe, o processo filho chama a execução de um programa por meio da chamada de sistema execve(). Por outro lado, a diferença é que o pai não aguarda o fim da execução do filho, afinal, ela deve ser em *background*. Por fim, o comando sohroda não imprime o código de saída do processo filho.

2 Exemplo de Execução

Para ver o programa funcionando, aqui está um exemplo de execução dentro do VirtualBox utilizando Minix:

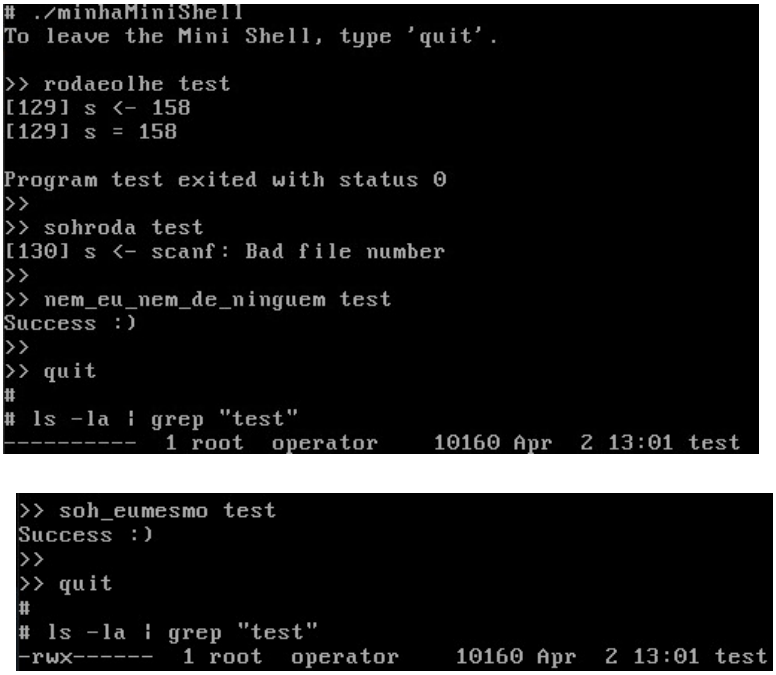

Figura 1: Execução das quatro chamadas de sistema.

Primeiramente o comando rodaeolhe é executado com o programa test. Esse programa apenas pede um input do usuário e depois o imprime. Vemos que o comando roda corretamente e imprime o código de saída 0, ou seja, que o processo filho terminou sem erros.

Logo após isso, o comando sohroda é chamado com o mesmo programa. Dessa vez, no entanto, há uma saída (esperada) de erro. Isso ocorre porque, com esse comando, o arquivo a ser executado não deve ter acesso ao teclado. No entanto, como ele pede um input do teclado do usuário, retorna-se um erro. Com isso, é possível assegurar que o teclado foi corretamente monopolizado pela shell.

Em seguida, é executado o comando nem_eu_nem_de_ninguem com o arquivo test. Nota-se que o processo foi bem-sucedido graças à mensagem "Success :)". A fim de assegurar essa correta execução, um ls -la é digitado para ver as permissões atuais do arquivo, que estão todas zeradas. Em outras palavras, a proteção 000 foi aplicada ao arquivo com sucesso.

Por último, o comando soh_eumesmo é feito no arquivo test. A mudança no arquivo é bem-sucedida e verificada com ls -la. Apenas o proprietário do arquivo passa a ter as permissões de leitura(r), escrita(w) e execução(x). Em outras palavras, a proteção 700 foi aplicada ao arquivo com sucesso.

3 Entrega

Este presente documento é o relatório do EP1 de MAC0422. Além dele, deve ser entregue uma imagem .ova contendo o executável da shell em /usr/local/bin e o programa fonte em C no diretório /usr/local/src. No entanto, a imagem .ova excede, em tamanho, o limite estabelecido pelo E-Disciplinas. Dessa forma, ela pode ser acessada pelo diretório no GitHub do EP ou por uma pasta no Google Drive. Abaixo, encontram-se os links:

- GitHub
- Google Drive