Sistemas Operacionais B - Engenharia de Computação

Projeto 2

Participantes	RA
Bruno Vicente Donaio Kitaka	16156341
Paulo Jansen de Oliveira Figueiredo	16043028
Rafael Fioramonte	16032708
Raissa Furlan Davinha	15032006
Vinicius Trevisan	16011231

Sumário:

1. Introdução:

Esse projeto teve o intuito de explorar em detalhes o desenvolvimento e aplicação de um módulo de kernel como um sistema de arquivo utilizando a API criptográfica do kernel Linux. O estudo visa o desenvolvimento do código do módulo, compilação do mesmo, instalação como novo módulo do kernel da máquina e utilização dele em um programa em espaço de usuário. Para podermos testar nosso módulo precisamos criar uma imagem de disco e montá-la em com um sistema de arquivo **minix**.

O módulo modificado recebe uma chave ao ser inserido no kernel em que cada dois dígitos hexadecimais correspondem a um **byte**, essa é utilizada para as funções de criptografia implementadas no mesmo. A fim de auxiliar as tarefas de teste, foi desenvolvido, um programa em espaço de usuário capaz de realizar operações de leitura e escrita em arquivos. Nesse programa, ao realizar uma chamada de sistema com a finalidade de escrever no arquivo, os dados são transmitidos para o módulo e inseridos de forma criptografada no mesmo, enquanto para a leitura, os dados são transmitidos para o módulo a fim de serem exibidos ao usuário de forma descriptografada.

A criptografia e descriptografia desenvolvida utilizam o algoritmo AES (*Advanced Encryption Standard*, uma especificação para criptografia de dados estabelecida pelo Instituto Nacional de Padrões e Tecnologia) em modo ECB (*Electronic CodeBook*, um modo de operação de criptografia simples não encadeada).

2. Fundamentação Teórica:

- **Criptografia e Descriptografia:** A criptografia consiste em converter informações comuns em texto não legível. A descriptografia, por outro lado, é o inverso, em outras palavras, passar o texto cifrado não reconhecido de volta a um texto reconhecível. Na maioria dos casos, uma chave conhecida é utilizada para realizar ambas as tarefas de modo a garantir a segurança das informações, essa, tendo que ser devidamente armazenada, para impossibilitar o acesso aos dados por terceiros e ser possível recuperar os mesmos.

- **Algoritmo AES:** AES entende-se como uma especificação para criptografia simétrica ou assimétrica podendo ser de 128, 192 ou 256 bits, esse sendo o tamanho máximo para a chave de entrada. É uma cifra por blocos, ou seja, a entrada e saída são limitadas conforme o tipo de especificação utilizada. Dessa forma, para realizar a criptografia ou descriptografia de uma grande quantidade de dados, este algoritmo deve ser executado iterativamente.

- **Sistema de arquivo:** Um sistema de arquivo é um conjunto de estruturas lógicas, ou seja, feitas diretamente via software, que permite ao sistema operacional ter acesso e controlar os dados gravados em qualquer dispositivo de armazenamento. Cada sistema operacional lida com um sistema de arquivos diferente e cada sistema de arquivos possui as suas peculiaridades, como limitações, qualidade, velocidade, gerenciamento de espaço, entre outras características. É o sistema de arquivos que define como os bytes que compõem um arquivo serão armazenados no disco e de que forma o sistema operacional terá acesso aos dados.

- **Sistema de Arquivo Minix:** desenvolvido por *Andrew S. Tanenbaum* com um objetivo de ser utilizado em seu sistema operacional denominado **minix**. Provou ser um sistema de arquivos muito problemático devido às limitações nele existentes já

que os arquivos não podiam ter nomes maiores do que 14 caracteres e as partições não podiam ser maiores do que 64 megabytes.

3. Desenvolvimento:

O código fonte do sistema Minix foi obtido a partir do repositório oficial do sistema operacional *linux* disponível na plataforma *gitHub* e utilizado como base para o módulo modificado. A fase inicial do projeto consistiu-se no estudo e entendimento do sistema de arquivos, de modo a encontrar os pontos chave das operações de escrita e leitura em um arquivo. Para isso foi necessário inicialmente modificar os códigos de arquivo Minix encontrados, colocando funções que, de certa forma, traçavam as chamadas realizadas *(printk)* para se entender o fluxo de funcionamento do sistema de arquivos.

Após o entendimento, fomos capazes de identificar as funções de escrita e leitura, sendo possível modificar seus ponteiros, interceptando-as, tudo isso partiu com o objetivo de incluir entre essas operações de arquivos as funções de criptografia antes dos dados serem escritos pela função *write* e a de descriptografia após a leitura pela função *read*.

As respectivas funções de escrita e leitura descritas no parágrafo acima foram encontradas dentro do arquivo ***file.c***, sendo que a estrutura ***minix_file_operations*** aponta para a função de leitura ***generic_file_read_iter*** e escrita ***generic_file_write_iter*** . Dentro da estrutura, essas funções genéricas foram substituídas por funções próprias (método *“man in the middle”*), ou seja, foram interceptadas de modo que antes de serem chamadas, as novas funções desenvolvidas eram chamadas em seu lugar, modificando os dados e então, os repassando para as genéricas.

Nas funções de interceptação de escrita e leitura desenvolvidas, o conteúdo é cifrado e decifrado, respectivamente, a partir do algoritmo de criptografia feita no projeto 1.

Por fim foi desenvolvido um programa em espaço de usuário capaz de realizar a leitura e escrita em um arquivo previamente especificado. Seu funcionamento inicia-se escolhendo a operação a ser realizada. Definimos **1** para criar um novo arquivo e **2** para atualizar um arquivo existente. Ao escolher a opção **1**, passa-se o caminho onde o arquivo será criado e uma *string* que, após recebida pelo programa em espaço de usuário, será gravada no arquivo pelo módulo criado. Por outro lado, ao escolher a opção **2**, passa-se também o caminho onde o arquivo será atualizado, exibe o conteúdo do mesmo e recebe uma *string* que reescreverá o conteúdo arquivo

Para compilação do módulo *minix* utilizamos o próprio *makefile* contido no código fonte do módulo executado por um *script*, esse *script* também é responsável por montar o arquivo imagem que utiliza o sistema de arquivo *minix*.

Na figura 1 mostramos como se deu o processo de compilação do módulo utilizando o comando ***make*** com o argumento ***“-C”*** que permite inserir o diretório cujo *Makefile* será lido, no caso, o diretório de *headers* do *kernel*, onde a versão é obtida pelo comando ***uname -r***. O parâmetro ***M*** é o diretório do módulo a ser compilado. Note que o segundo aviso que aparece durante o processos é devido ao manuseio do ponteiro ***iov_iter*** pelo fato da estrutura ***iov*** ser definida como constante.

Figura 1: Demonstração da compilação do módulo minix.

Na Figura 2, mostramos o passo a passo da criação da imagem de disco e sua montagem com o sistema de arquivos ***minix***. Primeiramente utilizamos o comando ***dd*** onde o parâmetro ***if*** é o diretório da imagem, ***of*** é o nome do arquivo, ***bs*** é o tamanho do bloco e ***count***, o número de blocos. Em seguida utilizamos o comando losetup que associa o arquivo criado a um dispositivo de bloco ***/dev/loop0***. O comando ***mkfs*** cria um sistema de arquivo no diretório ***/dev/loop0***, o parâmetro ***-t*** é o sistema de arquivos a ser utilizado, no caso, o *minix*. Então criamos um diretório ***/mnt/point1*** com o comando ***mkdir*** e, em seguida, montamos o dispositivo na pasta que criamos, utilizando o comando ***mount*** com o parâmetro ***-t*** (sistema de arquivos) [*minix*].

Figura 2: Exemplo de criação e montagem de uma imagem utilizando o sistema de arquivo minix.

Após a imagem ser devidamente montada, inserimos o módulo que foi compilado anteriormente (como na Figura 1) utilizando o comando ***insmod***, passando como parâmetro para o módulo a **chave de criptografia** (*key="abcd"*). O comando ***lsmod*** é utilizado para a listagem dos módulos instalados e o ***grep*** filtra a saída com a palavra que definimos, de modo que possamos verificar se o módulo foi realmente inserido.

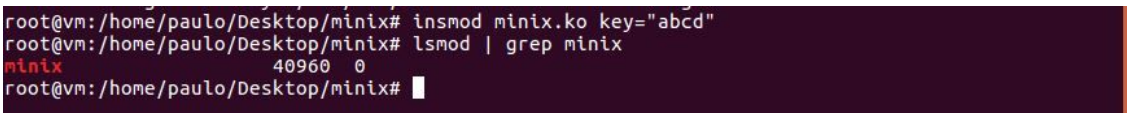

Figura 3: Exemplo de inserção do módulo.

Com o comando ***dmesg*** é possível visualizar o *log* do *kernel*, ou seja, as mensagens do módulo recém instalado.

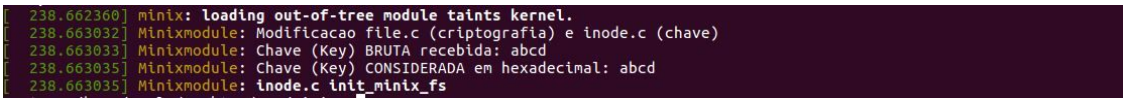

Figura 4: Confirmação do carregamento e chave recebida pelo módulo.

4. Resultados:

As Figura 5 e 6 exemplificam a execução do programa em espaço de usuário a partir da opção **1** (Criação de um novo arquivo) como também seu menu (descrito na seção 3).

Figura 5: Interface do programa de teste em espaço de usuário para a opção 1.

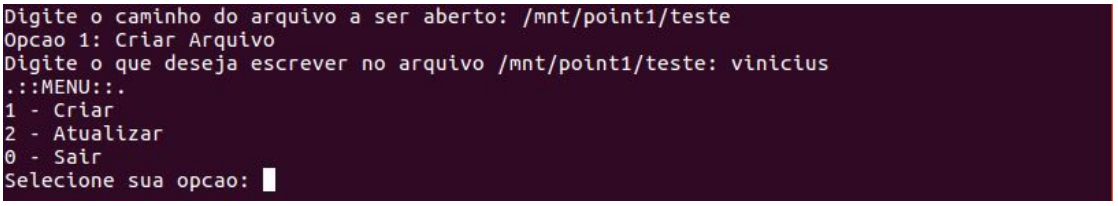

Figura 6: Exemplo da criação do arquivo “teste”com o conteúdo ”vinicius”.

Após criado o arquivo com a string fornecida pela opção 1, testamos a leitura do mesmo utilizando o comando ***hd,*** que basicamente exibe o conteúdo do arquivo em formato hexadecimal, com ambos os módulos minix a fim de testar se função de leitura estava realmente descriptografando os dados.

Figura 7: Teste da leitura do arquivo com o módulo minix modificado (dados descriptografados).

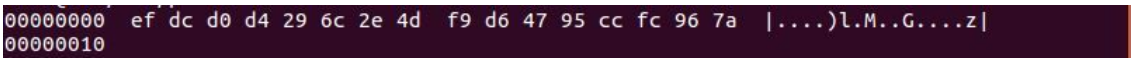

Figura 8: Teste da leitura do arquivo com o módulo minix original (dados criptografados).

As Figuras 9, 10, 11, 12 e 13 exibem o resultado do teste realizado a partir da opção **2** (atualização) no programa em espaço de usuário. Inicialmente utilizamos o módulo modificado (Figuras 9 10 e 11) e logo após, o mesmo teste com o módulo original (Figuras 12 e 13).

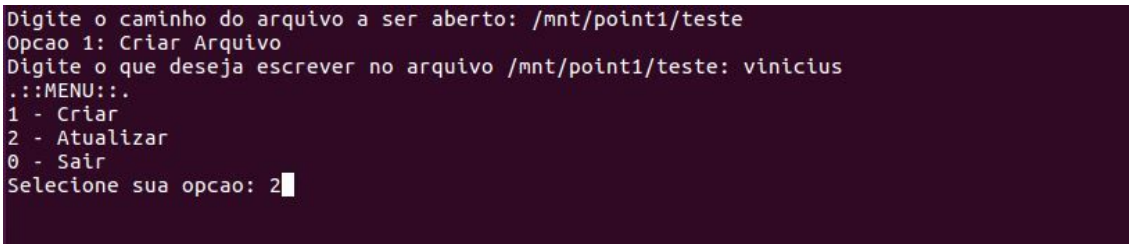

Figura 9: Interface do programa de teste em espaço de usuário para a opção 2.

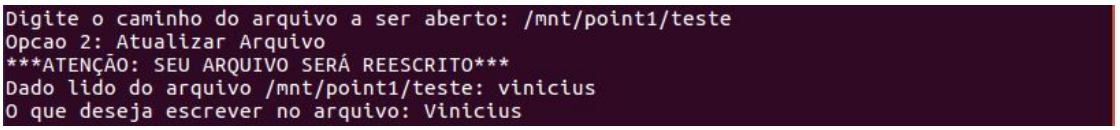

Figura 10: Exemplo da atualização do arquivo "teste" com a palavra "Vinicius" utilizando o módulo minix modificado.

Figura 11: Exemplo de leitura do arquivo "teste" criptografado utilizando o módulo minix modificado.

Figura 12: Exemplo da atualização do arquivo "teste" criptografado utilizando o módulo minix original, note que a exibição dos dados é ilegível.

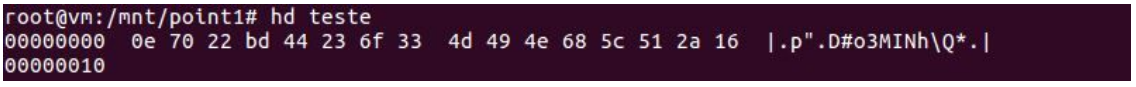

Figura 13: Exemplo de leitura do arquivo "teste" criptografado utilizando o módulo minix original.

5. Conclusão

Após testes alternando entre o código minix modificado e o minix sem alterações, os resultados demonstraram que a criptografia e descriptografia estão sendo realizadas com sucesso. Desta forma, podemos concluir que o módulo modificado está funcionando corretamente como descrito no enunciado do projeto.

Contudo, cabe a observação de que durante o processo de descriptografia por blocos de 16 bytes é impossível definir o tamanho do texto nele contido, daí a fato de ser necessário a utilização da técnica chamada ***padding*** que consiste no preenchimento das posições vazias do bloco com o valor "**0**" como demonstrado na Figura 11.

Concluindo, após esse projeto ficou claro o funcionamento básico de um sistema de arquivos, sua importância dentro de um sistema operacional e os passos necessários para interceptar funções de leitura e escrita. Outro fato importante a ser exposto é a importância de sistemas que implementam técnicas de criptografia a fim de proteger seus dados contra acessos não autorizados.