CEATEC – Centro de Ciências Exatas, Ambientais e de Tecnologia

Faculdade de Engenharia de Computação

SISTEMAS OPERACIONAIS B

Projeto 2

João Pedro Favara RA:16061921

Murilo Martos Mendonça RA:16063497

Victor Hugo do Nascimento RA:16100588

Campinas 26 de Novembro de 2018

Sumário

1. Introdução

A criptografia é um importante recurso em um sistema. Trata-se de uma ferramenta de segurança extremamente valiosa que permite a proteção da informação e garante que apenas quem pode saber o conteúdo de uma mensagem tenha capacidade de entendê-lo. A aplicação da criptografia em uma camada baixa do sistema, como um driver, traz mais facilidade no desenvolvimento de sistemas abordando a segurança da informação. Deste modo propôs-se um projeto no qual um driver de dispositivo criptográfico deveria ser construído e acessível ao usuário através de um programa.

Este documento relata este projeto, da matéria de Sistemas Operacionais B, o qual determina o desenvolvimento de um sistema que realize a integração de um módulo de kernel que utilize a API de criptografia criptografar os arquivos criados ou editados pelo usuário. A atividade determina a criação, compilação e execução correta de um módulo de kernel que utilize a API de criptografia do kernel do Linux. Deveriam ser utilizados as funções de encriptação e decriptação com o algoritmo *AES*.

2. Objetivo

A atividade objetiva o desenvolvimento de um sistema de módulo de kernel que utiliza a API de criptografia do kernel Linux juntamente com um programa de usuário capaz de realizar chamadas ao driver. O módulo por sua vez, possuem as funções que utilizávamos no arquivo localizado no módulo */dev/crypto*, utilizado no primeiro projeto, ou seja, o sistema se trata de um driver de dispositivo que capaz de encriptar e decriptar os arquivos criados ou editados pelo usuário.

3. Detalhes de projeto

O sistema proposto foi desenvolvido em máquina virtual sob a ferramenta virtual box. A versão de Kernel utilizada é a 4.15.0-39-generic, já utilizado e compilado nas atividades anteriores ao projeto.

O módulo desenvolvido consegue encriptar e decriptar corretamente os arquivos, no entanto demonstra a decriptação e encriptação no log do kernel, e no arquivo, fica apenas a frase criptografada.

O sistema tem como finalidade fazer a função de um driver de dispositivo criptográfico

(crypto device driver) capaz de receber requisições e enviar respostas através do arquivo, foi construído e testado por cada componente em separado.

No arquivo file.c que é um arquivo do minix system file tivemos que alterar duas chamadas de funções utilizadas por ele, a de leitura e a de escrita, intituladas agora como read_sob e write_sob. Em read_sob chamamos nossa função de decriptação e em write_sob chamamos a encriptação. O grupo não conseguiu manipular os arquivos de forma que fosse passada a key de criptografia, então foi definida uma key default para o system: “ABCDEF0123456789”.

Criptografía e Descriptografia

O módulo de criptografia foi desenvolvido em espaço de kernel como proposto, este

contém uma função para cifrar um buffer utilizando o algoritmo AES da **API Crypto**, passando para ela uma chave (o qual não é determinada na inserção do módulo) e que é utilizado na criptografia.

Também contém em seu escopo uma função para decifrar um dado que foi criptografado pela função citada anteriormente. A função de decriptografar recebe o buffer lido do arquivo e devolve para o usuário a mensagem original.

4. Resultados obtidos

A seguir são mostrados os resultados obtidos no experimento.

4.1 Carregando o módulo do Minix

Nesta etapa carregamos o módulo Minix do kernel, utilizado no formato do dispositivo de armazenamento.

4.2. Dispositivo de armazenamento com formato Minix

```
1  #!/bin/bash
2  echo "criador de sistemas de arquivos"
3  dd if=/dev/zero of=file.img bs=1k count=10000
4  echo "transformando file.img em um arquivo de bloco"
5  losetup /dev/loop16 file.img
6  echo "montando sistema de arquivo minix no arquivo file.img"
7  mkfs.minix -c /dev/loop16 10000
```

Figura 1 - Depois de carregado o módulo que o dispositivo de armazenamento utiliza para manipular arquivos, é montada a unidade de armazenamento “loop16”. Todos os arquivos que forem criados ou movidos para essa pasta automaticamente utilizarão as funções write_sob e read_sob que estão implementadas no módulo minix.ko

4.3. Criptografia de um arquivo

Com o comando **dmesg** no terminal conseguimos obter os *printk’s* gerados pelo módulo no momento em que criamos um arquivo dentro do dispositivo. Este módulo está utilizando o sistema de arquivos minix. A partir desses prints podemos observar que a função para criptografar foi chamada, retornando o tamanho da string a ser escrita, a string em si, e o resultado da criptografia realizada, tudo isso para podermos analisar seu funcionamento.

```
[  869.477619] FUNCAO WRITE
[  869.477624] size de from->iov->iov_base:10
[  869.477632] buf com from->iov->iov_base:teste sob
[  869.477655] Crypted: 24 9a a9 d3 10 5b e9 4d 2e 40 f1 78 7e 04 18 32 fd b6 72
 4a 73 79 d0 18 4a 32 04 10 02 0f 51 63
```

Figura 2 – Acima é mostrada a frase “teste sob” e abaixo o conjunto de caracteres que representam a mesma frase depois da criptografia.

5. Conclusão

O sistema desenvolvido proporcionou um desafio de lidar com uma API que possui uma documentação escassa para a versão do kernel utilizada, no entanto, aos poucos foi possível notar a semelhança na utilização da versão mais antiga com a versão mais atual. Houve uma grande dificuldade para encontrar o local correto para se colocar as funções de criptografia e descriptografia e fazer com que funcionassem corretamente como o esperado, então, o grupo ficava toda hora realizando o insmod e rmmod para testes, porém o grupo atingiu o objetivo de encriptar a mensagem passada através de uma string de entrada, mas infelizmente, não conseguimos exibir a mensagem corretamente dentro do arquivo de texto, feito pelo usuário. Os desafios encontrados permitiram um maior conhecimento sobre o kernel do sistema e os problemas encontrados na integração.

6. Referências

[1] Linux Kernel Crypto API:. Disponível em. https://www.kernel.org/doc/html/v4.12/crypto/index.html

[2] Writing a Linux Kernel Module - Part 2: A Character Device.Disponivel em :http://derekmolloy.ie/writing-a-linux-kernel-module-part-2-a-character-device

[3] The Linux Kernel Module Programming Guide. Peer Jay Salzman. J. P, Burian.M, Pomerantz .O.Copyright © 2001 Peter Jay Salzman.2007−05−18 ver 2.6.4

Linux Kernel Crypto API: