Projeto 1

## Sistemas Operacionais B

**Nome: RA:**

Leonardo Carbonari 13126578

Matheus Franceschini 13129788

Pedro Tortella 13035555

Tales Falcão 13146394

Diogo Esteves Furtado 15153927

Kaíque Ferreira Fávero 15118698

### Introdução:

O projeto 1 de Sistemas Operacionais B teve como objetivo principal implementar um módulo de Kernel que utilizasse a API criptográfica do Linux para realizar três funções: Criptografar um dado fornecido pelo usuário, utilizando o algoritmo de criptografia AES (***Advanced Encryption Standard***), decriptografar o dado que foi criptografado anteriormente e calcular o resumo criptográfico (*hash*) de um dado fornecido pelo usuário utilizando a função criptográfica SHA1 (***Secure Hash Algorithm***).

### Detalhes de projeto do módulo:

O módulo foi criado utilizando como base o módulo fornecido pelo professor através do link: "http://derekmolloy.ie/writing-a-linux-kernel-module-part-2-a-character-device". Este módulo cria um arquivo no diretório **/dev**, o qual é utilizado para interagir com o driver. Na função que inicializa o módulo é criado este arquivo e, além disso, foi adicionado um parâmetro *key* para receber por parâmetro a chave criptográfica que será utilizada. O código para realizar essa operação foi, também utilizado na segunda atividade da matéria:

**module\_param(key, charp, 0000);**

**MODULE\_PARM\_DESC(key, "key");**

Após armazenada a chave, é criado o driver que servirá como interface entre o usuário e o módulo implementado. Para que essa comunicação aconteça, o usuário deve realizar operações de escritas e leituras sobre este dispositivo seguindo as seguintes regras:

* Para criptografar uma string, deve-se escrever “**c** **dado\_a\_ser\_criptografado**”;
* Para decriptografar o dado que foi previamente criptografado, deve-se escrever “**d dado\_a\_ser\_criptografado**”;
* Para calcular o resumo criptográfico de uma string, deve-se escrever “**h dado\_a\_ser\_calculado**”.

Para selecionar qual opção será escolhida pelo usuário foi implementada uma função que é chamada toda vez que ocorre uma escrita no arquivo **/dev/crypto**. Essa função separa o primeiro caractere recebido (operação) dos demais (dados). Após isso, é feito um *switch case* e é chamada a função referente a criptografia, decriptografia e cálculo do resumo criptográfico.

A seguir serão explicadas as implementações de cada uma das operações:

* Criptografar: Para criptografar o dado recebido, existe uma função *“encrypt”* que recebe uma sequência de caracteres. Este método consiste, basicamente, de quatro chamadas:
  + crypto\_alloc\_cipher("aes", 0, 16): Esta chamada à Crypto API retorna uma estrutura que permite tratar uma cifra para um único bloco usando a criptografia 'AES';
  + crypto\_cipher\_setkey(tfm,key,16): Esta chamada à crypto API vincula uma chave *'key'* à estrutura passada em seu primeiro parâmetro. Como segundo e terceiros parâmetros são passadas a chave e seu tamanho, respectivamente. O grupo tomou a liberdade de definir os tamanhos das chaves sempre em 16 bytes;
  + crypto\_cipher\_encrypt\_one(tfm,c,in): Esta chamada à crypto API criptografa um bloco usando o algoritmo e a chave especificada na estrutura passada como primeiro parâmetro. O segundo e o terceiro parâmetro são, respectivamente, o destino e a origem dos dados.
  + crypto\_free\_cipher(tfm): Esta chamada à Crypto API libera a estrutura passada como primeiro parâmetro.

A chamada ao método que realiza a criptografia é feita em um laço repetitivo, pois, caso a mensagem a ser encriptada seja maior que a chave, este passo deve ser feito n vezes, até que a mensagem inteira seja encriptada.

* Decriptografar: Para decriptografar o dado recebido, existe uma função *“decrypt”* que opera sobre a mensagem encriptada, salva em uma variável global. Este método consiste, basicamente, de três chamadas:
  + crypto\_alloc\_cipher("aes", 0, 16): Esta chamada à Crypto API retorna uma estrutura que permite tratar uma cifra para um único bloco usando a criptografia 'AES';
  + crypto\_cipher\_setkey(tfm,key,16): Esta chamada à crypto API vincula uma chave *'key'* à estrutura passada em seu primeiro parâmetro. Como segundo e terceiros parâmetros são passadas a chave e seu tamanho, respectivamente. O grupo tomou a liberdade de definir os tamanhos das chaves sempre em 16 bytes;
  + crypto\_cipher\_decrypt\_one(tfm,c,encrypted): Esta chamada à crypto API decriptografa um bloco usando o algoritmo e a chave especificada na estrutura passada como primeiro parâmetro. O segundo e o terceiro parâmetro são, respectivamente, o destino e a origem dos dados.

A chamada ao método que realiza a tradução do dado criptografado é feita em um laço repetitivo, pois, caso a mensagem encriptada seja maior que a chave, este passo deve ser feito n vezes, até que a mensagem inteira seja traduzida.

* Calcular Hash (SHA1): Para calcular o resumo criptográfico existe uma função “hash” que recebe como parâmetro um vetor de caracteres e foi utilizada a a seguinte implementação:
  + Primeiramente é inicializada uma estrutura do tipo crypto\_shash utilizando a função *crypto\_alloc\_shash* que recebe “sha1” como primeiro parâmetro, definindo qual será o tipo de hash.
  + Após isso é alocada a memória para a estrutura do tipo shash\_desc.
  + Por fim, é utilizado a função crypto\_shash\_digest com os parâmetros desc (contendo uma flag e a estrutura crypto\_shash), in (dado de entrada), len (tamanho do dado de entrada) e sha1res que é um vetor que recebe o resultado do cálculo do sha1.

Por fim, foi feito um programa de usuário com a finalidade de testar o módulo desenvolvido. Nele é inserido um texto contendo a operação que deseja fazer e o dado que será passado (se existir dado) e, após apertar ENTER, recebe o resultado do que foi calculado no módulo.

O programa de testes foi desenvolvido a partir de um programa contido no Material Complementar disponibilizado pelo professor, através do link: "<http://derekmolloy.ie/writing-a-linux-kernel-module-part-2-a-character-device>" (Listing 4).

Foram feitas adaptações no código:

* Apresentação de uma interface que indica o padrão de entrada para cada função do módulo, **“operador dado”** no caso de Cifra e Cálculo do Hash, ou somente **“operador”** para a função Decifrar.
* Mensagens de erro no caso de falha ao abrir o arquivo, na escrita e na leitura do resultado do dispositivo.
* Alteração no diretório de manipulação do arquivo, para a localização do arquivo de dispositivo do módulo ,**“*/dev/crypto* “**.

### Resultados obtidos:

Mesmo com diversos desafios encontrados durante o desenvolvimento do projeto, o grupo obteve ótimos resultados, os quais estão demonstrados a seguir:

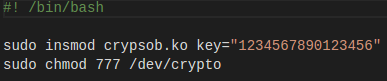


Figura 1. Arquivo de inicialização do módulo.

Para facilitar o desenvolvimento do projeto, bem como os seus testes, o grupo criou um arquivo com extensão *.sh*, que insere o módulo criado, com a chave mostrada na Figura 1, e também fornece as todas as permissões para o arquivo do módulo (como exibido na mesma imagem).

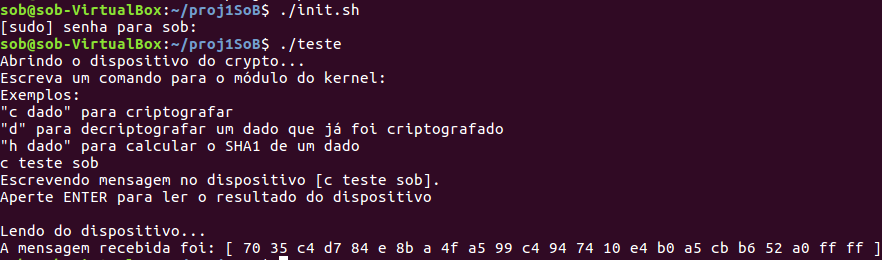


Figura 2. Teste de criptografia

A Figura 2 mostra a execução do programa de teste em espaço de usuário, utilizando a opção de criptografar (“c”) e o dado (“teste sob”). O resultado da criptografia pode ser visto na última linha da imagem, no formato hexadecimal.

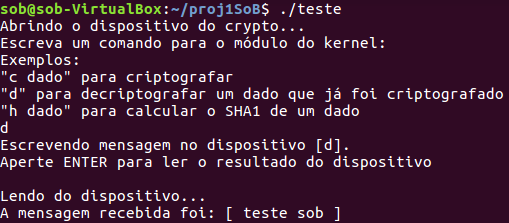


Figura 3. Teste de decriptografia

Na Figura 3 é possível observar a execução do programa de teste descriptografando o dado inserido criptografado na Figura 2. Para isso, o comando necessário foi apenas a opção de descriptografar, uma vez que o dado já estava salvo no dispositivo. O resultado da operação pode ser visto na última linha da imagem.

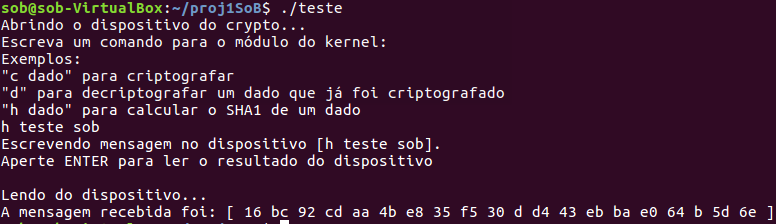


Figura 4. Cálculo do resumo criptográfico (SHA1)

Na Figura 4, é exibida a execução do programa teste para se calcular o resumo criptográfico do dado fornecido. Para isso, a opção “h” foi utilizada, com o dado “teste sob”. O resultado do cálculo pode ser visto na última linha, em formato hexadecimal. Para verificar se o resultado do resumo está correto, o grupo visitou o site http://www.sha1-online.com/ e inseriu o mesmo texto usado no programa de testes, para que o texto fosse transformado em uma cadeia hexadecimal e o grupo pudesse compará-lo com o resultado obtido pelo módulo. Abaixo é mostrada a Figura 5, a qual consta o resultado obtido no site citado acima:

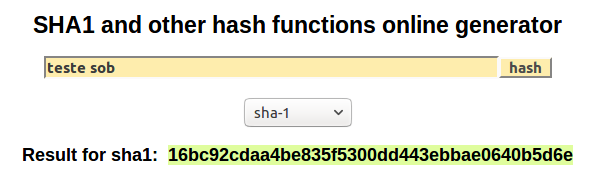


Figura 5. Resultado do resumo criptográfico no site http://www.sha1-online.com/

### Conclusão:

Após finalizar o desenvolvimento do projeto, o grupo concluiu que esse projeto foi desafiador, pois além de trabalhar com a criação de um módulo de *kernel*, também teve que lidar com a *API* criptográfica fornecida pelo *kernel* do Linux, sendo que a mesma mudou consideravelmente entre versões diferentes do kernel.

Além disso, foi necessário utilizar o conhecimento adquirido nas atividades desenvolvidas anteriormente a fim de desenvolver o módulo de *kernel* proposto.