

*Sistemas Operacionais B*

**Projeto 1  
*Crypto Device Driver***

Breno Baldovinotti | RA: 14315311

Caroline Gerbaudo Nakazato | RA: 17164260

Marco Antônio de Nadai Filho | RA: 16245961

Nícolas Leonardo Külzer Kupka | RA: 16104325

*24/10/2019*

*CEATEC – Centro de Ciências Exatas, Ambientais e de Tecnologia*

**1.Introdução**

A criptografia é de extrema importância no mundo atual, fornecendo segurança e confiabilidade às constantes trocas de informações. No Linux, não é diferente. Sua API criptográfica é amplamente utilizada em outras porções de kernel, e é fundamental para o correto funcionamento do sistema operacional, pois permitindo uma comunicação correta e segura com outros dispositivos ou até mesmo entre os próprios mecanismos do SO.

Este projeto visa a análise e o entendimento das técnicas de implementação de um módulo de kernel para o Linux que faz uso de sua API criptográfica presente no mesmo. Também, busca familiarizar-se com os detalhes de implementação de tal módulo de kernel e suas dependências.

Para isto, foi desenvolvido um módulo de kernel Linux versão de um dispositivo criptográfico, capaz de cifrar e decifrar mensagens através do algoritmo AES em modo CBC, utilizando-se de uma chave e um vetor de inicialização fornecidos na inserção do módulo. O dispositivo também deveria ser capaz de calcular o hash de uma string através do algoritmo SHA1.

Para o teste deste dispositivo, foi desenvolvido um programa em espaço de usuário que se comunica com o mesmo, podendo abrir o dispositivo, enviar uma requisição e exibir a resposta retornada ao usuário.

**2. Detalhes do Projeto**

O módulo foi desenvolvido na versão 4.1X do Kernel Linux, estando localizado no arquivo de dispositivo /dev/crypto. Seu Major Number é alocado dinamicamente. Quando o módulo do kernel for carregado, é necessário informar o parâmetro “key” (uma chave simétrica) e o parâmetro iv (vetor de inicialização), os quais são utilizados pelo algoritmo AES CBC para cifrar e decifrar os dados. Ambos os parâmetros correspondem a uma string representada em hexadecimal. O algoritmo de criptografia AES possuí um tamanho de bloco fixo em 128 bits, mas apresenta inúmeras variantes. Neste projeto, foi utilizada a variante AES CBC 128, o qual apresenta uma chave de comprimento de 128 bits.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

A carga do módulo executada é:

insmod cryptomodule.ko key=”0123456789ABCDEF” iv=”0123456789ABCDEF”

O primeiro caractere define a operação, sendo elas cifrar (c), decifrar (d) e calcular o resumo criptográfico (h).

O dado é passado em formato de string representados em hexadecimal (cada byte corresponde a dois dígitos hexa).

A operação de cifrar, retorna uma string correspondendo aos dados fornecidos durante a requisição, cifrados com o algoritmo AES em modo CBC utilizando-se a chave fornecida durante a carga do módulo, representados em hexadecimal.

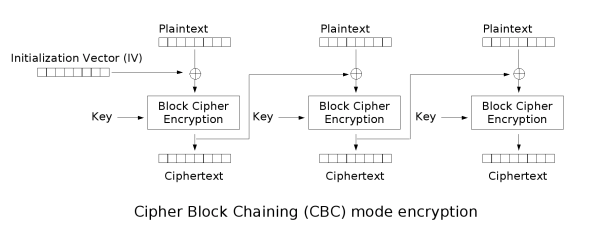


Figura 1: Diagrama do algoritmo de cifra AES em modo CBC.

Já a operação de decifrar, a resposta será uma string correspondendo aos dados fornecidos durante a requisição representados em hexadecimal, decifrados com o algoritmo AES em modo CBC utilizando-se a chave fornecida durante a carga do módulo.

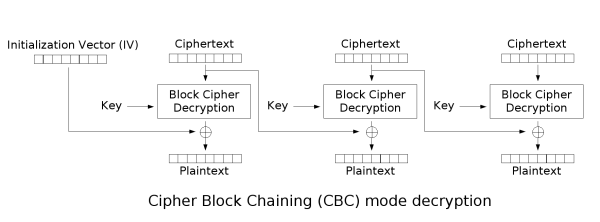
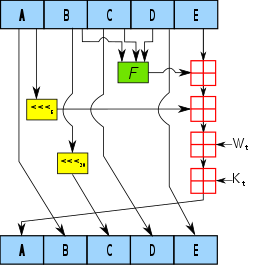


Figura 2: Diagrama do algoritmo de decifra AES em modo CBC.

Na operação de cálculo de hash, é retornado uma string correspondendo ao resumo criptográfico em hexadecimal dos dados fornecidos durante a requisição, utilizando-se o algoritmo SHA-1.



Uma iteração dentro da função de compressão do SHA-1:

A, B, C ,D e E são palavras de 32 bits do estado;

F é uma função não-linear que varia;

<<< denota a rotação de bits à esquerda em n espaços, n varia para cada operação;

Wt é a palavra da mensagem expandida da rodada t;

Kt é a constante da rodada t;

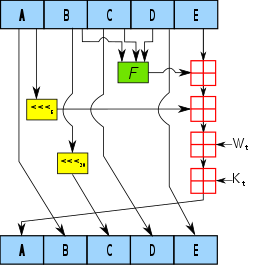
 denota uma adição módulo 232.

Figura 3: Diagrama do algoritmo de hash SHA-1.

Também foi desenvolvido no módulo de kernel o MUTEX LOCKS para bloquear um processo no espaço do usuário, caso o dispositivo /dev/crypto estiver sendo utilizado por outro processo.

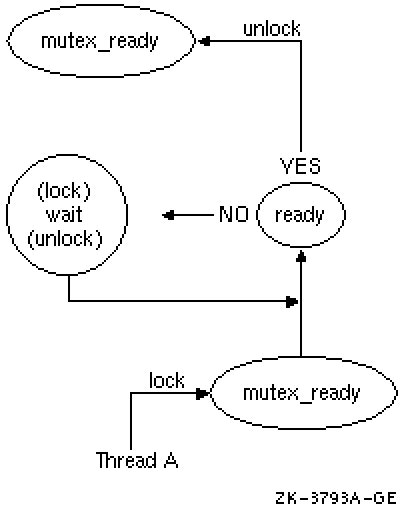


Figura 4: Diagrama do algoritmo do MUTEX LOCKS.

**3.Resultados**

* Makefile passando os valores de key e iv

Figura 5: Output do Makefile passando os valores de iv e key.

* Cifrar

Figura 6: Output da operação de cifra do módulo.

* Decifrar

Figura 7: Output da operação de decifra do módulo.

* Hash

Figura 8: Output da operação de hash do módulo.

* Mutex

Figura 9: teste do MUTEX LOCKS no módulo.

**5.Conclusão**

Através deste projeto, foi possível familiarizar-se com os detalhes de implementação de um módulo de kernel que faz uso da API criptográfica do kernel Linux.

Através do estudo deste módulo foram aprimorados os conceitos de implementação, compilação, instalação e teste de um novo módulo de kernel que realiza as funções de cifrar, decifrar e calcular o hash dos dados fornecidos pelo usuário.